НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
(повна назва інституту/факультету)

Кафедра автоматики та управління в технічних системах
(повна назва кафедри)

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ О.І. Ролік
(підпис) (ініціали, прізвище)
“___”_____20__ р.

Магістерська дисертація

зі спеціальності (спеціалізації) 121 Інженерія програмного забезпечення
(код і назва спеціальності)

на тему: Система управління режимами функціонування електричної мікромережі

Виконав: студент __6__ курсу, групи ____ІТ-з83МП_____
(шифр групи)

Упіров Іван Сергійович _____ _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник _____проф., д.т.н., с.н.с. Чемерис О. А.___ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2019 року

Національний технічний університет України
«Київський політехнічний інститут
імені Ігоря Сікорського»

Факультет (інститут) Факультет інформатики та обчислювальної техніки
(повна назва)

Кафедра Автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121 Інженерія програмного забезпечення
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ _____
(підпис) (ініціали, прізвище)
«___»_____20__ р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Упіров Іван Сергійович
(прізвище, ім'я, по батькові)

1. Тема дисертації Система управління режимами функціонування електричної мікромережі

науковий керівник дисертації Чемерис О.А., д.т.н. с.н.с
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження моделювання роботи електричної мікромережі

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) система керування режимами електричної мікромережі

5. Перелік завдань, які потрібно розробити дослідження і аналіз існуючих систем моделювання електричних мікромереж; розробка мікросервісної архітектури; розробка системи керування електричною мікромережею; інтеграція системи з блокчейном; створення інтерфейсу користувача; тестування розробленої системи

6. Орієнтовний перелік ілюстративного (графічного) матеріалу сценарії використання системи, структурна схема системи, структура модулів системи, діаграми послідовностей основних функцій системи

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

9. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів магістерської дисертації	Примітка
1	Аналіз існуючих рішень	30.09.2019 – 06.10.2019	
2	Підбір оптимальних технологій	07.10.2019 – 13.10.2019	
3	Розробка архітектури системи	14.10.2019 – 20.10.2019	
4	Аналіз методів взаємодії з блокчейном	21.10.2019 – 27.10.2019	
5	Розробка серверної частини системи	28.10.2019 – 03.11.2019	
6	Тестування системи	18.11.2019 – 24.11.2019	
7	Оформлення пояснювальної записки	25.11.2019 – 01.12.2019	
8	Подача дисертації на перевірку	03.12.2019	

Студент _____ _____
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ _____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

РЕФЕРАТ

Дисертація освітньо-кваліфікаційного рівня “магістр” на тему «Система управління режимами функціонування електричної мікромережі»: 118 с., 5 рис., 68 табл., 8 додатків, 36 джерел.

Об’єкт дослідження – система для моделювання роботи електричної мікромережі та керування режимами роботи об’єктів моделі на основі технології блокчейн.

Мета роботи – розробити систему моделювання роботи електричної мікромережі та керування нею. Проаналізувати існуючі системи моделювання електричних мереж. Запропонувати власну реалізацію збереження даних про модель електричної мікромережі у системі блокчейн.

Наукова новизна дослідження полягає у тому, що імплементовано систему яка надає можливість симулювати роботу електричної мікромережі та керувати режимами роботи її компонентів та зберігає дані моделі у системі блокчейн.

При розробці системи та її компонентів використовувалися сучасні технології програмування, такі як Python, Flask, Ethereum, SQL Server.

Прогнозні припущення про розвиток дослідження – застосування системи керування режимами електричної мікромережі у системах моделювання та аналізу електричних мереж.

REPORT

Master's dissertation on subject on subject "Electric microgrid management system" contains 118 pages, 5 pictures, 68 tables, ?? sources.

The object of study is an electric microgrid management system, based on the blockchain technology.

The purpose of the work is to develop an electric microgrid management system, to analyze the existing similar systems and to propose the new implementation of storing of the microgrid data in the blockchain system.

The scientific novelty of the research is that the electric microgrid management system uses the integration with the blockchain engine to store the dynamic data of the model.

The system was implemented with the modern programming technologies such as Python, Flask, Ethereum, SQL Server.

Predictive assumptions about the development of the study - the integration of the electric microgrid management system to the systems for the modelling and analysis of electrical networks.

ЗМІСТ

1. АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

1.1 Актуальність систем моделювання електричних мікромереж

При високих темпах розвитку економіки обсяги електроспоживання в світі з 1990 до 2019 року зросли більш ніж в два рази [1]. Забезпечення таких рівнів росту виробництва електроенергії в майбутньому неможливе без створення нової технологічної основи енергетики. Один з варіантів розвитку цієї сфери інженерії - використання принципів інтелектуальних електричних мереж, або мікромереж [2]. Тому в даній роботі розглядається питання дослідження особливостей функціонування мікромереж за допомогою комп'ютерного моделювання.

На рисунку 1.1 зображено діаграму росту обсягу електровикористання електроенергії в світі.

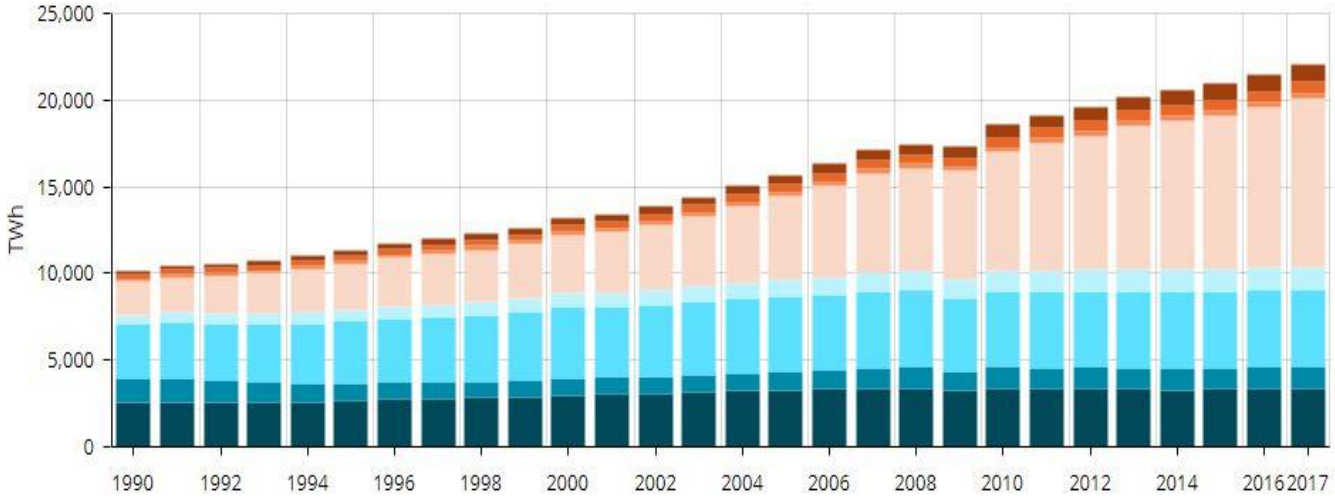

Рисунок 1.1 - Обсяги електропостачання в світі

Стрімкий темп росту енергоспоживання у світі створює необхідність оптимізації роботи енергетичних систем. Для того, щоб підтримувати швидкість росту виробництва товарів та наукового прогресу на поточному рівні, необхідно впроваджувати нові підходи в процес розробки електричних мереж.

Одним з таких підходів є комп’ютерне моделювання системи електропостачання. Цей підхід дозволяє проаналізувати поведінку системи у штучно створених умовах, провести навантажувальне тестування та відслідкувати зміни стану системи після функціонування протягом великих проміжків часу.

Комп’ютерне моделювання системи електропостачання може значно пришвидшити процес розробки системи та зробити її тестування дешевшим та легшим в реалізації.

Створена модель електричної мікромережі може бути повторно використана у разі необхідності внесення змін до існуючої системи, чи розгортання подібної системі в інших умовах.

1.2 EnergyPlus

EnergyPlus - комплексна програма для моделювання використання електроенергії в закритих електричних мережах, яку інженери, архітектори та науковці використовують для створення моделей, що дозволяють аналізувати особливості роботи опалення, охолодження, вентиляції, освітлення і технологічних навантажень [3].

Основні функції програми:

- інтегроване моделювання роботи системи ОВК для аналізу умов термічного стану системи, швидкості реагування системи та її навантажувального тестування;
- засноване на тепловому балансі рішення для моделювання та аналізу променистих і конвективних впливів на систему і розрахунки конденсації;
- погодинний аналіз взаємодії між тепловими зонами і системами опалення, вентиляції і кондиціонування.

Це дозволяє EnergyPlus моделювати системи з швидкою динамікою, а також знижувати швидкість моделювання з метою збільшення точності.

EnergyPlus - це консольна програма, яка зчитує ввід і записує висновок в текстові файли. Вона поставляється з рядом утиліт, включаючи IDF-Editor для створення вхідних файлів з використанням простого інтерфейсу, схожого на електронну таблицю, EP-Launch для управління файлами вводу і виводу і виконання пакетного моделювання, а також EP-Compare для графічного порівняння результатів двох або більше моделювання. EnergyPlus працює в операційних системах Windows, Mac OS X і Linux.

1.3 OpenDDS

OpenDSS - це комплексний інструмент моделювання систем електропостачання. Він підтримує створення моделей практично всіх частотних діапазонів, які необхідні для аналізу систем розподілу електроенергії. Крім того він дозволяє будувати моделі з урахуванням нових критеріїв, розроблених для задоволення майбутніх потреб, пов'язаних із інтелектуальними мережами, модернізацією існуючих мереж та дослідженнями відновлюваної енергії. Інструмент OpenDSS застосовується з 1997 року для підтримки різноманітних науково-дослідних та консалтингових проектів, що потребують аналізу розподілених систем електропостачання [4].

Багато функцій інструменту спочатку були призначені для створення моделей для аналізу розподіленої генерації електроенергії для широкого вжитку. Зараз це найбільш розповсюджена сфера використання OpenDSS. Також інструмент дозволяє досліджувати енергоефективність систем електропостачання. OpenDSS розроблений таким чином, щоб його можна було нескінченно масштабувати та легко модифікувати для задоволення майбутніх потреб.

1.4 Порівняння

В ході роботи над дисертацією було розроблено систему управління режимами функціонування електричної мікромережі. Створена система дозволяє створювати моделі електричних мікромереж Smart Grid та досліджувати їх роботу.

Під час розробки системи було вирішено наступні задачі:

- створено комп'ютерну модель мікромережі;
- проаналізовано проекти використання технології блокчейн для управління мікромережами в світі;
- розроблено програмну систему управління комп'ютерною мережею з подобовою реєстрацією даних моніторингу;
- доведено спроможність атоматизації управління мікромережею.

Для постійного збереження даних у розробленій системі було використано Ethereum - розподілену обчислювальну платформу на основі технології блокчейн, що дозволяє зберігати кожну операцію, що відбувається в системі та верифікувати цілісність даних. Крім того, це створює можливість горизонтального масштабування системи через те, що різні частини мікромережі можуть моделюватися на різних обчислювальних машинах, дані з яких будуть синхронізуватись у спільному децентралізованому сховищі. За допомогою використання механізму смарт-контрактів, ключові операції системи можуть виконуватись у середовищі Ethereum, що дозволяє відстежувати усі зміни її стану. Таким чином модель предметної області може бути інкапсульована у сховищі даних.

Система має консольний інтерфейс, що дозволяє автоматично виконувати моделювання системи у разі зміни вхідних даних. Результати моделювання можуть бути експортовані в структурованому текстовому вигляді для подальшого аналізу та візуалізації.

Створена система є кросплатформенною і працює на операційних системах Windows, Mac OS X і Linux.

На таблиці 1.1 зображено порівняння функціоналу розробленої системи з найбільш схожими існуючими системами. Для порівняння обрані системи EnergyPlus та OpenDDS. Перша є пропрієтарною системою, друга являє собою бібліотеку для розробки подібних рішень.

Таблиця 1.1 - Порівняння з існуючими рішеннями

	Модель	EnergyPlus	OpenDDS
Можливість гнучкої конфігурації параметрів мережі	+	+	+
Візуалізація зібраних даних	+	+	-
Експорт зібраних даних для подальшої обробки сторонніми системами	+	-	+
Можливість паралельного запуску великої кількості моделей	+	-	-
Підтримка кластеризації та розподілених обчислень	+	-	-
Гарантування цілісності даних	+	-	-
Інтерфейс для конфігурування за допомогою терміналу	+	-	-

1.5 Висновок розділу

Значне збільшення темпів споживання електроенергії створює необхідність пошуку та впровадження нових підходів у проектуванні та розробці електромереж.

Для оптимізації процесів проектування та тестування електромереж може бути використано комп'ютерне моделювання. EnergyPlus та OpenDDS – приклади систем, що дозволяють створювати моделі електричних мереж. Система, розроблена в ході роботи над дисертацією, дозволяє ефективно моделювати електричну мікромережу, здійснювати раціональне та оптимальне керування моделлю та аналізувати її поведінку. Система створена на основі технології блокчейн з використанням смарт-контрактів, що дозволяє здійснювати горизонтальне масштабування, гарантує цілісність даних та надає інструменти для постійного моніторингу її стану.

2. ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

1.1 Функціональні вимоги

2.1.1 Обробка транзакцій

При моделюванні електричної мікромережі необхідно реалізувати можливість об’єктів предметної області обмінюватись повідомленнями. Отримані повідомлення можуть змінювати стан об’єктів, чи ініціювати певну їх поведінку. Для реалізації деяких правил предметної області необхідно реалізувати можливість групування кількох повідомлень в одну логічну операцію. Наприклад:

- додавання нового джерела електричної енергії в мережу, оновлення стану усіх потенційних споживачів;
- перебудова маршруту доставки електричної енергії, зміна стану мережі, зміна режиму роботи джерел енергії.

Система повинна реалізувати функціонал відстеження усіх транзакцій, що відбулись у моделі від початку її роботи.

2.1.2 Бізнес правила

В доменній області системи наявні наступні бізнес-правила:

- модель електричної мікромережі може складатись з двох та більше вузлів, з’єднаних між собою;
- модель електричної мікромережі може бути в одному з наступних станів: “нормальний”, “аварійний”;
- під час запуску модель знаходиться в “нормальному” стані;

- вузол може бути джерелом електричної енергії, споживачем, чи з’єднувальним елементом;
- з’єднувальні елементи використовуються для з’єднання окремих ланок мережі з метою формування найбільш оптимальних маршрутів доставки електричної енергії;
- кожен вузол належить певній ланці мережі;
- адміністратор системи може змінювати значення атрибутів моделі в ручному режимі;
- джерело електричної енергії має наступні атрибути: середня продуктивність, поточна продуктивність, максимальна продуктивність;
- споживач електричної енергії має наступні атрибути: середнє споживання, поточне споживання, максимальне споживання;
- електрична мікромережа може бути розширена вузлом будь-якого типу під час роботи;
- джерело електричної енергії може бути в одному з наступних станів: “активне”, “відключене”, “на обслуговуванні”;
- коли джерело має стан “відключене”, чи “на обслуговуванні”, його продуктивність дорівнює нулю;
- коли джерело переходить у стан “відключене”, чи “на обслуговуванні”, система перевіряє, чи має кожна ланка мережі достатньо джерел, щоб забезпечити усіх споживачів;
- якщо в результаті аналізу результатів перевірки виявляється, що в системі існують ланки, яким недостатньо джерел, щоб забезпечити усіх споживачів, ланки повинні бути автоматично перебудовані таким чином, щоб у кожній ланці вироблялось достатньо електричної енергії;

- якщо в результаті аналізу виявляється, що навантаження на ланки розподілено нерівномірно, ланки повинні бути автоматично перебудовані таким чином, щоб навантаження на джерела електричної енергії було розподілено рівномірно;
- якщо в результаті аналізу результатів перевірки виявляється, що система не може потрапити у такий стан, щоб у кожній її ланці усі споживачі могли отримувати необхідну їм кількість електричної енергії, модель повинна переходити у аварійний стан;
- для виводу системи з аварійного стану адміністратор системи повинен в ручному режимі додати нові джерела електричної енергії, або зменшити їх кількість, чи обсяг використання.

2.1.3 Формування звітів

Під час функціонування моделі адміністратор повинен мати змогу експортувати стан кожного об’єкту доменної моделі в текстовому форматі. Експортований документ повинен містити дату експорту, загальний стан моделі, стан кожного об’єкта моделі та значення всіх атрибутів кожного зареєстрованого у системі об’єкта в структурованому вигляді. Експортований звіт може бути використаний для аналізу та візуалізації даних.

Адміністратор повинен мати змогу обрати для експорту певні ланки системи, що має бути відображено у звіті.

Звіти повинні експортуватися в тому ж процесі, в якому працює програма, не перериваючи її виконання. Одночасно може експортуватися лише один звіт. Адміністратор повинен мати змогу формувати чергу експорту звітів про різні ланки системи.

2.1.4 Адміністративні функції

Адміністратор системи конфігурує модель мікромережі перед її запуском. Конфігурація може бути описана за допомогою текстового файлу, що імпортується системою, чи напряму через інтерфейс системи. У разі імпорту з файлу адміністратор має змогу скоригувати імпортований опис моделі через інтерфейс системи.

Під час роботи моделі система повинна надавати адміністратору можливість керувати моделлю електромережі в ручному режимі.

Під час роботи модель повинна надавати адміністратору можливість моніторингу внутрішнього стану системи. Цей функціонал повинен бути реалізований через збереження логів у реальному часі та виведення найбільш важливих повідомлень на консоль адміністратора. Також адміністратор системи повинен мати змогу експортувати звіт про стан моделі під час її роботи, чи після зупинки.

2.1.5 Рівні авторизації

Система повинна мати чотири рівні авторизації:

- спостерігач - має доступ до працюючої моделі через інтерфейс користувача, він не може керувати моделлю в ручному режимі, чи змінювати її стан;
- працівник - має доступ до користувацького інтерфейсу системи та може керувати моделлю в ручному режимі, у користувача на цьому рівні авторизації є можливість експортувати дані про модель, але він не може змінювати її стан;
- адміністратор інфраструктури - має повний доступ до всіх віртуальних машин, на яких працює система та до системи блокчейну;
- глобальний адміністратор - включає в себе повноваження усіх попередніх рівнів авторизації та має повний доступ до всіх модулів системи, може керувати моделлю в ручному режимі, чи змінювати її стан.

2.1.6 Зовнішні залежності

У таблиці 2.1 наведено список усіх програмних зовнішніх залежностей системи та короткий опис кожної з них.

У таблиці 2.2 наведено список усіх інфраструктурних зовнішніх залежностей системи та короткий опис кожної з них.

Таблиця 2.1 - Програмні зовнішні залежності системи

№	Назва	Опис
1	Web3	Бібліотека для мови програмування Python, що дозволяє взаємодіяти з системами, побудованими на базі технології Ethereum.
2	Flask	Бібліотека для мови програмування Python, що дозволяє будувати веб-додатки.
3	Gunicorn	Інтерфейс для взаємодії з веб-сервером.
8	Autopep8	Лінтер.

Таблиця 2.2 - Інфраструктурні зовнішні залежності системи

№	Назва	Опис
1	Ethereum	Платформа для створення децентралізованих онлайн-сервісів на базі технології блокчейн.
2	Geth	Імплементація платформи Ethereum.

Продовження таблиці 2.2

№	Назва	Опис
3	Git	Система контролю версій, що використовується в процесі компіляції системи Geth.
4	Golang	Мова програмування, на якій імплементовано Geth. Необхідна для компіляції системи Geth.
5	Mingw	Набір вільного програмного забезпечення для розробки Windows додатків. Необхідна для компіляції системи Geth.

2.2 Нефункціональні вимоги

2.2.1 Доступність

Уся інформація, що демонструється користувачу не в текстовому вигляді, повинна мати текстову альтернативу, що виконує еквівалентну ціль – представлення продукту.

Дані програми можуть бути представлені користувачеві різними способами без втрати інформації чи структури. Інформація, що демонструється користувачеві, може бути описана програмно чи доступна у текстовому вигляді. Коли послідовність, у якій представлена інформація, впливає на її значення чи сприйняття, коректна послідовність демонстрації інформації визначається програмно. Для зручності інформація може бути подана у книжній чи альбомній орієнтації.

Важливим є візуальне оформлення продукту, яке досягається різними шляхами. За рахунок розділення переднього плану та фону користувачеві легше концентруватись на найбільш важливих елементах керування. Але колір не може бути єдиним

візуальним засобом передачі даних, оскільки користувачі можуть мати вади сприйняття кольору чи інші особливості. Окрім кольору для кращого сприйняття потрібно використовувати форматування елементів керування та тексту. Візуальне представлення тексту та зображень має коефіцієнт контрастності не менше ніж 4.5:1. Розмір тексту може бути оптимізовано та кореговано без залучення допоміжних засобів, без втрати інформації, чи функціоналу. Контент також може бути переформатовано без втрати інформації, чи функціоналу.

Усі функції системи мають бути доступні для виклику з клавіатури. Користувач повинен мати достатньо часу для реакції на зміни стану системи. Користувач повинен розуміти способи навігації та пошуку контенту.

Дані, що демонструються користувачеві, мають бути читабельні та зрозумілі. Коли будь-який компонент отримує фокус, він не ініціює зміну контексту. Зміна налаштувань будь-якого компонента інтерфейсу не викликає автоматичну зміну контексту. Навігаційні механізми, що повторюються у системі, завжди виникають у одному відносному порядку. У разі введення некоректних даних, система повинна сигналізувати про це. Вікно з описом помилки повинно бути інформативним та зрозумілим. Якщо помилка автоматично виявляється і система може запропонувати пропозиції щодо виправлення, користувач отримує можливість вибору одного з запропонованих варіантів.

Експортовані дані про стан моделі повинні бути достатньо цілісними, раціонально структурованими, щоб їх можна було інтерпретувати за допомогою зовнішніх систем для візуалізації даних.

2.2.2 Відновлення даних

У разі введення користувачем некоректних даних, система повинна мати змогу відмінити всі транзакції, що відбулися після її переходу у некоректний стан.

2.2.3 Розширюваність

Система повинна бути реалізована таким чином, щоб у разі необхідності було можливо легко реагувати на нові обставини і вносити у програмний код зміни, які б дозволили оптимізувати роботу моделей, додавати нові режими та будувати більш складні моделі електричних мікромереж, що складаються з компонентів з більшою кількістю атрибутів. Також алгоритм балансування навантаження на ланку мікромережі повинен бути абстрагованим від системи таким чином, щоб у разі потреби його було можливо легко замінити на інший.

2.2.4 Відмовостійкість

У випадку виведення з ладу одного чи кількох вузлів блокчейну система повинна продовжувати моделювати електричну мікромережу у звичайному режимі. Якщо система працює у розподіленому режимі та моделює роботу різних ланок однієї мережі на різних машинах, у разі виходу з ладу одного чи кількох вузлів кластеру, система повинна сигналізувати про це користувачеві, але продовжувати функціонування ланок мікромережі на працюючих вузлах. У разі апаратного збою сховища даних система повинна зберегти дані на іншому вузлі кластера, якщо це можливо.

2.2.5 Сумісність

У таблиці 2.3 наведено опис сумісності системи з операційними системами. У таблиці 2.4 наведено опис сумісності системи з інфраструктурними залежностями.

Таблиця 2.3 - Сумісність з операційними системами

№	Назва	Версія
1	Windows	7, 8, 8.1, 10
2	Windows Server	2008, 2008 R2, 2012, 2012 R2, 2016, 2019
3	Ubuntu	14.04.6, 16.04.6, 18.04.3, 19.10
4	Debian	6.0, 7.0, 8.0, 9.0, 10.0
5	CentOS	7.*, 8.*

Таблиця 2.4 - Сумісність з інфраструктурними залежностями

№	Назва	Версія
1	Geth	1.9.7
2	Nginx	1.17.5
3	Docker	18.09
4	Python	3.8.0

У таблиці 2.5 наведено опис сумісності системи з програмними залежностями.

Таблиця 2.5 - Сумісність з програмними залежностями

№	Назва	Версія
1	Web3.py	5.2.2
2	Flask	1.1.1
3	Gunicorn	19.9.0
4	Autopep8	1.4.4

Клієнтська частина системи імплементована за допомогою веб-технологій і виконується у браузері користувача.

У таблиці 2.6 описано сумісність системи з браузерами.

Таблиця 2.6 - Сумісність з браузерами

№	Назва	Версія
1	Internet Explorer	9.0 +
2	Mozilla Firefox	3.5 +
3	Google Chrome	3.0 +
4	Safari	3.1 +
5	Microsoft Edge	20.1 +
6	Opera	10.5 +

2.2.6 Підтримуваність

Кодова база системи повинна бути імплементована таким чином, щоб її можна було підтримувати, корегувати, та у разі необхідності вносити зміни в її логіку. Зокрема до підтримуваності системи висуваються наступні вимоги:

- іменування програмних структур повинно відображати їх сутність;
- повторне використання модулів системи;
- кожен клас повинен мати одну чітко визначену відповідальність;
- система повинна бути відкрита для розширення, але закрита для модифікації;
- диспетчеризація поведінки повинна відбуватись за рахунок динамічної типізації та поліморфізму;
- кожен модуль системи повинен залежати лише від тих об’єктів, які йому справді потрібні;
- напрямок залежностей у системі повинен бути протилежним напрямку виконання;
- кожна підсистема повинна бути максимально самодостатньою;
- кожна підсистема повинна бути реалізована максимально просто;
- система повинна бути покрита тестами;
- зміни, внесені до однієї підсистеми, не повинні порушувати роботу іншої підсистеми;
- у разі виникнення помилки, вона повинна бути виправлена за допомогою внесення змін у якомога меншу кількість місць;
- побудова середовища для розробки та тестування повинна бути швидкою;
- код не повинен повторюватися;
- код повинен бути написаний таким чином, щоб його було легко читати.

2.2.7 Масштабованість

Система повинна бути реалізована таким чином, щоб її можна було масштабувати як вертикально, так і горизонтально. У випадку, якщо уся модель симулюється на одному комп’ютері, система повинна вигравати у продуктивності у разі збільшення кількості обчислювальних ресурсів. Також система повинна працювати у кластерному режимі і покращувати свою продуктивність у разі збільшення кількості вузлів кластера.

2.2.8 Безпека

Система повинна гарантувати, що неавторизовані користувачі не зможуть отримати доступ до даних моделі. Авторизовані користувачі отримують права у відповідності до своєї ролі в системі. У разі пошкодження даних моделі вони повинні бути повернуті до останнього корректного стану.

2.2.9 Стабільність

Під час роботи моделі система не повинна аварійно завершувати свою роботу, дані роботи моделі не повинні бути втрачені. Час симуляції не обмежено, система повинна зберігати стальки даних симуляції, який об’єм сховища буде до неї під'єднаний.

2.2.10 Здатність до тестування

Модулі системи повинні мати одну чітку відповідальність та буди слабо зв’язаними один з одним, щоб систему можно було покрити модульними тестами.

Також середовище для тестування системи повинно швидко розгортатися для тестування в ручному режимі та інтеграційних тестів.

2.3 Висновок розділу

До розробленої системи було висунуто наступні функціональні вимоги:

- обробка транзакцій;
- бізнес правила;
- формування звітів;
- адміністративні функції;
- рівні авторизації;
- перелічені залежності.

Також до неї було висунуто наступні нефункціональні вимоги:

- доступність;
- відновлення даних;
- розширюваність;
- відмовостійкість;
- сумісність;
- підтримуваність;
- масштабованість;
- безпека;
- стабільність;
- здатність до тестування.

3. СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

3.1 Основні сценарії використання системи

У додатку Б наведено основні сценарії використання системи, доступні ролі, що має найбільш широкі повноваження у системі - Адміністратор.

У таблиці 3.1 наведено опис кожного сценарію використання системи.

Таблиця 3.1 - Сценарії використання системи

№	Назва	Опис
1	Створення моделі	Адміністратор вводить у систему дані про електричну мікромережу, модель якої необхідно побудувати.
2	Конфігурація ланки мікромережі	Під час роботи системи адміністратор у ручному режимі здійснює налаштування властивостей ланки електричної мікромережі, робота якої моделюється.
3	Конфігурація вузла мікромережі	Під час роботи системи адміністратор у ручному режимі здійснює налаштування властивостей вузла електричної мікромережі, робота якої моделюється.
4	Повернення моделі у попередній стан	Якщо модель опиняється у некорректному стані, адміністратор має можливість повернути її до будь-якого попереднього стану.
5	Експорт даних	Під час роботи моделі, чи після завершення процесу моделювання адміністратор має можливість експортувати дані про модель для їх подальшої обробки та візуалізації.

3.1.1 Сценарій створення моделі

Користувач, що має роль адміністратора системи, вводить у систему дані про електричну мікромережу, модель якої необхідно побудувати. Модель може бути створено програмними засобами - у цьому разі адміністратор за допомогою інтерфейсу системи описує усі ланки та вузли системи, та задає їх параметри. Також адміністратор має можливість імпортувати в систему раніше створену модель. Можливий імпорт з локального файлу та з мережевого сховища, що надає мережевий програмний інтерфейс, сумісний з системою. Після імпорту моделі її можна відредагувати та запустити у виконання. В додатку Е наведено діаграму послідовності цього сценарію.

3.1.2 Сценарій конфігурації ланки мікромережі

Під час управління моделлю в ручному режимі користувач має змогу працювати з ланками електричної мікромережі. Користувач може створювати нові ланки з нових, чи існуючих вузлів та додавати їх до моделі. Крім того користувач може додавати чи видаляти вузли ланок, що вже існують у системі.

3.1.3 Сценарій конфігурації вузла мікромережі

Під час управління моделлю в ручному режимі користувач має змогу працювати з вузлами електричної мікромережі. Користувач може додавати нові вузли до існуючих ланок, чи модифікувати стан, чи властивості роботи існуючих вузлів. Користувач може працювати як з джерелами енергії, так і зі споживачами. Властивості джерела енергії, якими може керувати користувач:

- режим роботи;
- продуктивність;

- ланка, що обслуговується джерелом.

Властивості споживача енергії, якими може керувати користувач:

- режим роботи;
- об’єм використання енергії;
- ланка, до якої входить користувач.

3.1.4 Сценарій повернення моделі у попередній стан

Адміністратор системи має можливість повернути систему у стан, у якому вона була у будь-який момент часу своєї роботи. Також система може бути повернута у стан, який вона отримала після виконання будь-якої транзакції. В додатку Є наведено діаграму послідовності цього сценарію.

3.1.5 Сценарій експорту даних

Адміністратор системи має можливість експортувати дані про модель електричної мікромережі під час функціонування моделі, або після завершення роботи системи. Дані можуть бути експортовані у локальний файл, або у мережеве сховище, що надає мережевий програмний інтерфейс, сумісний з системою.

3.2 Висновок розділу

Адміністратор системи має доступ до наступних сценаріїв використання системи:

- створення та повернення у попередній стан та моделі;
- конфігурація ланки та вузла мікромережі;
- експорт даних.

4. СТРУКТУРНА СХЕМА СИСТЕМИ

4.1 Основна структурна схема

У додатку В наведено основну структурну схему системи.

Клієнти взаємодіють з системою через веб-інтерфейс. Для цього імплементовано веб-сервіс. Опис основних функцій веб-сервіса представлено у таблиці 4.1.

Усі внутрішні компоненти системи з’єднані між собою у віртуальну приватну мережу, що дозволяє їм безпечно обмінюватись повідомленнями навіть у разі розгортання у різних дата-центрах. Веб-сервіс відправляє усі отримані та оброблені повідомлення на модуль, який керує моделюванням електричної мікромережі. Даний модуль містить у собі більшу частину доменної бізнес-логіки та оркеструє роботу усієї системи. У разі необхідності збереження даних, що не мають прямого відношення до роботи моделі електричної мікромережі, таких, як, дані про користувачів системи, налаштування, тощо, Оркестратор передає необхідне повідомлення модулю взаємодії з базою даних. Модуль взаємодії з базою даних інкапсулює шар репозиторіїв, які дозволяють зберігати у базі всі необхідні доменні об’єкти, а також модифікувати, читати та видаляти їх. У разі отримання повідомлень про необхідність зміну стану моделі, Оркестратор передає ці повідомлення модулю взаємодії з блокчейном. Даний модуль викликає функції смарт-контракту, що виконується інфраструктурою блокчейну та імплементує симуляцію подій, що відбуваються під час роботи електричної мікромережі. У разі необхідності система може працювати з додатковими вузлами блокчейну, що дозволяє створювати більше точок верифікації даних. Навантаження автоматично балансується між усіма вузлами блокчейна, щоб підтримувати достатній час відповіді системи.

Таблиця 4.1 - Опис основних функцій веб-сервіса

№	Назва	Опис
1	Ідентифікація	Процедура розпізнавання користувача в системі.
2	Аутентифікація	Процедура встановлення належності користувача системі за його унікальним ідентифікатором.
3	Авторизація	Процедура надання користувачеві повноважень у системі та доступу до захищених ресурсів, згідно з його повноваженнями.
4	Формування повідомлень доменної області на основі веб-запитів	Веб-сервер, що обслуговує зовнішній трафік, перенаправляє усі коректні запити на веб-сервіс. На основі типів запитів, значень заголовків та тіл запитів веб-сервіс створює повідомлення предметної області та передає їх наступним модулям у системі.
5	Маршрутизація запитів користувача	У залежності від запиту користувача веб-сервіс ініціює запуск різних сценаріїв функціонування системи.
6	Форматування відповідей на запити	Коли у процесі обробки запиту користувача відбувається подія, про яку необхідно сповістити його - веб-сервіс формує відповідь.

4.2 Схема блокчейн інфраструктури

Взаємодія будь-якого компонента з розподіленою блокчейн інфраструктурою відбувається через драйвер Web3 та EVM.

Web3 - це спеціальна бібліотека, що реалізує можливість взаємодії з системою Ethereum через gRPC протокол [5]. EVM - уніфікована точка управління розподіленою мережею вузлів блокчейну [6]. За допомогою Web3 можно обмінюватись повідомленнями з EVM і таким чином взаємодіяти з блокчейном, зокрема публікувати смарт-контракти та викликати їх функції. EVM гарантує цілісність даних між усіма вузлами системи та балансує навантаження на вузли [7]. У ролі користувача у даному випадку виступає модуль взаємодії з блокчейном.

4.3 Організація компонентів у системі

У додатку Г наведено візуалізацію компонентів системи.

Найбільш важливий компонент Веб-Сервісу - RequestHandler. Цей компонент описує мережевий інтерфейс для усіх можливих випадків взаємодії користувача з системою. Призначення даного компонента - прийняти запит від користувача, визначити сценарій використання системи, до якого отриманий запит відноситься та делегувати обробку запиту компоненту Orchestrator, що є частиною модуля керування моделюванням електричної мікромережі.

У таблиці 4.2 описано усі доменні об’єкти, які створює модуль RequestHandler.

Модуль Orchestrator отримує повідомлення доменної моделі від модуля RequestHandler та інкапсулює логіку їх обробки. Для цього Orchestrator взаємодіє з двома модулями: PersistenceEngine та BlockchainEngine. Для збереження налаштувань, внесених користувачем, використовується компонент постійного збереження даних, а для взаємодії з моделлю електричної мікромережі відправляється повідомлення на

компонент взаємодії з блокчейном. Обидва ці модулі є частинами інших модулів, тому взаємодія здійснюється через мережеве з’єднання.

Таблиця 4.2 - Доменні об’єкти, які створює модуль RequestHandler

№	Назва	Опис
1	SmartGridModel	Описує мікромережу, робота якої моделюється.
2	Subnet	Описує ланку мікромережі.
3	Node	Описує вузол мікромережі.
4	ProviderNode	Розширює об’єкт Node та описує вузол мікромережі, що є джерелом електричної енергії.
5	ConsumerNode	Розширює об’єкт Node та описує вузол мікромережі, що є споживачем електричної енергії.
6	MediatorNode	Розширює об’єкт Node та описує вузол мікромережі, використовується для з’єднання ланок.

Модуль взаємодії з базою даних складається з чотирьох компонентів:

- компонент для обробки зовнішніх запитів - виконує функцію отримання та розпізнавання повідомлень від інших підсистем. На основі отриманих повідомлень даний компонент ініціює виклики відповідних методів компоненту Одиниця роботи.
- одиниця роботи - компонент для зв’язування компонентів репозиторіїв між собою [8].
- репозиторій налаштувань - компонент для збереження, читання, редагування та видалення даних налаштувань у постійному сховищі [9].

- репозиторій знімків системи - компонент для збереження, читання, редагування та видалення даних налаштувань у постійному сховищі.

Модуль BlockchainEngine реалізує логіку взаємодії системи з інфраструктурою блокчейну. У таблиці 4.3 наведено перелік та опис усіх компонентів, з яких складається модуль.

Таблиця 4.3 - Опис компонентів модуля BlockchainEngine

№	Назва	Опис
1	RequestHandler	Виконує функцію отримання та розпізнавання повідомлень від інших підсистем.
2	UnitOfWork	Компонент для зв’язування компонентів репозиторіїв між собою
3	NetworkRepository	Компонент для збереження даних ланок мережі у постійному сховищі
4	NodeRepository	Компонент для збереження даних вузлів мережі у постійному сховищі

4.4 Висновок розділу

Основні компоненти, з яких складається система:

- клієнтський застосунок
- веб-сервер
- веб-сервіс
- віртуальна приватна мережа, що з’єднує усі вузли системи
- модуль керування моделюванням

- модуль взаємодії з базою даних
- база даних
- модуль взаємодії з блокчейном
- вузли блокчейну

У таблиці 4.4 наведено опис усіх доменних об’єктів системи.

Таблиця 4.4 - Доменні об’єкти

№	Назва	Опис
1	Електрична мікромережа	Коренева доменна сутність обмеженого контексту, що описує мікромережу, робота якої моделюється.
2	Ланка мережі	Підмережа, що описує найбільш оптимальний шлях доставки електричної енергії від джерел до споживачів.
3	Вузол мережі	Об’єкт мережі, що приймає участь у енергетичній взаємодії.
4	Джерело енергії	Вузол мережі, що виробляє електричну енергію та обслуговує одну чи декілька ланок.
5	Споживач енергії	Вузол мережі, що споживає електричну енергію однієї чи кількох ланок.
6	З’єднуючий вузол	Вузол мережі, що використовується для формування ланок.

5. ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

5.1 Мови програмування

5.1.1 Python

Усі модулі системи, за винятком смарт-контрактів та елементів інтерфейсу користувача, розроблено за допомогою мови програмування Python. У таблиці 5.1 наведено перелік її ключових характеристик, що роблять її оптимальною для розробки даної системи.

Таблиця 5.1 - Основні характеристики мови програмування Python

№	Назва	Опис
1	Високорівнева	Забороняє ручне керування виділенням та звільненням пам’яті, що дозволяє зосередитись на розробці логіки системи.
2	Інтерпретована	Не потребує довгого та ресурсоємного етапу компіляції.
3	Мультипарадигменна	Дозволяє одночасно використовувати прийоми об’єктно-орієнтованого та функціонального програмування, коли це необхідно. Це дозволяє досягти більш виразного коду.

Крім того мова програмування Python є динамічно типізованою, що дозволяє не описувати типи, коли це не потрібно. Безкоштовно розповсюджуються інтерпретатори

цієї мови програмування для найбільш уживаних операційних систем. Python з’явився у 1990 році і за час свого існування отримав зручний інструментарій, менеджер залежностей та підтримку основних систем постійної інтеграції. Вихідний код інтерпретатора та базової бібліотеки є відкритим [10].

5.1.2 Solidity

Solidity - мова програмування, на якій було реалізовано смарт-контракт у системі Ethereum. Це повна за Тюрингом об’єктно орієнтована високорівнева мова програмування, що є частиною екосистеми Ethereum. Написану програму необхідно скомпілювати та розгорнути у середовищі блокчейну. Виконанням програми керує середовище Ethereum Virtual Machine. Воно забезпечує розподілене виконання обчислень, балансування навантаження та синхронізацію інформації між вузлами системи [11]. Основні можливості мови програмування Solidity:

- створення смарт-контрактів;
- використання директив компілятора;
- створення змінних;
- створення функцій;
- створення подій;
- створення переліків;
- керування рівнями доступу;
- збереження стану.

5.1.3 Javascript

Javascript - динамічна, мультипарадигменна, прототипна мова програмування, що використовується для реалізації сценаріїв веб-сторінок клієнтської частини системи.

Реалізує логіку інтерактивного відображення стану системи та асинхронного обміну даними з сервером [12]. У таблиці 5.2 наведено перелік її ключових характеристик, що роблять її оптимальною для розробки даної системи.

Таблиця 5.2 - Основні характеристики мови програмування Python

№	Назва	Опис
1	Переносимість	Єдина мова програмування, що підтримується усіми популярними браузерами.
2	Прототипність	Дозволяє реалізувати повторне використання коду за рахунок клонування існуючого примірника об’єкта - прототипу. Це дозволяє обмежити використання наслідування та підтримувати більшу інкапсуляцію.
3	Асинхронність	Механізм Event Loop дозволяє асинхронно взаємодіяти з користувачем, не “заморожуючи” елементи керування системою.
4	NPM	Розвинута система управління залежностями, що дозволяє повторно використовувати код, використовуючи його, як сторонню залежність.
5	Наявність фреймворків	Існує велика кількість фреймворків для побудови складних систем за допомогою мови Javascript. Такі рішення легше масштабувати та підтримувати.

5.2 Фреймворки

5.2.1 Flask

Flask - мінімалістичний фреймворк для мови програмування Python, що дозволяє будувати веб-застосунки WSGI. Він дозволяє легко почати роботу та дає змогу поступово розширювати її. Flask пропонує свій підхід до розробки веб-додатків, однак не зобов’язує його притримуватись та не вносить додаткових залежностей у систему [13]. Основні функції, що реалізує фреймворк Flask:

- робота у режимі налагодження;
- маршрутизація запитів;
- обробка запитів до статичних файлів;
- рендеринг шаблонів;
- доступ до даних запиту;
- перенаправлення запитів;
- обробка помилок;
- управління сесіями користувача;
- логування;
- взаємодія з WSGI Middleware;
- використання розширень;
- возгортання у сервері веб-застосунків.

5.2.2 React

React - відкритий Javascript фреймворк для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки. React дозволяє розробникам створювати великі веб-застосунки, які використовують дані,

котрі змінюються з часом, без перезавантаження сторінки [14]. У таблиці 5.3 наведено особливості даного фреймворку, які було враховано під час вибору його для реалізації системи керування режимами електричної мікромережі.

Таблиця 5.3 - Основні характеристики фреймворка React

№	Назва	Опис
1	Одностороння передача даних	Властивості передаються в рендерер компоненту, як властивості html тега. Компонент не може напряму змінювати властивості, що йому передані, але може їх змінювати через callback функції. Такий механізм називають «властивості донизу, події нагору».
2	Віртуальний DOM	React підтримує віртуальний DOM, що дозволяє бібліотеці визначити, які частини DOM змінилися, порівняно зі збереженою версією віртуального DOM, і таким чином визначити, як найефективніше оновити DOM браузера. Таким чином програміст працює зі сторінкою, вважаючи що вона оновлюється вся, але бібліотека самостійно вирішує які компоненти сторінки треба оновити.
3	JSX	Компоненти React зазвичай написані на JSX. Код написаний на JSX компілюється у виклики методів бібліотеки React. Розробники можуть так само писати на чистому JavaScript.
4	Методи життєвого циклу	Методи життєвого циклу – це різні методи, які вбудовуються за допомогою ReactJS. Вони дозволяють розробнику обробляти дані в різних точках життєвого циклу програми React.

5.3 Бібліотеки

5.3.1 Web3

Web3 - колекція бібліотек, що реалізують можливість взаємодії з локальною, чи віддаленою системою Ethereum, за допомогою використання з'єднання HTTP чи RPC. Web3 – єдиний офіційно сертифікований інструмент для взаємодії з системою Ethereum [5].

5.3.2 Gunicorn

Gunicorn (Green Unicorn) - WSGI сервер для мови програмування Python. Worker-процеси сервера працюють на основі системних викликів типу fork на процесі сервера. Gunicorn підтримує сумісність з великою кількістю веб-фреймворків, просто реалізований, не вимагає багато ресурсів для роботи та достатньо швидкий для даної системи. Gunicorn WSGI сервер необхідний для паралельної обробки запитів користувача у багатьох потоках та процесах [15].

5.3.3 Autopep8

Autopep8 - системна бібліотека, що автоматично форматує код, написаний на мові програмування Python, у відповідності до стандарту PEP8. Вона використовує утиліту pycodestyle, щоб визначити, які частини коду необхідно переформатувати. Autopep8 реалізує функціонал для виправлення більшості проблем форматування, о яких може повідомити pycodestyle. Використання бібліотеки Autopep8 допомагає тримати кодову базу у кращому косметичному стані, гарантуючи, що форматування всіх файлів з кодом буде однаковим [16].

5.4 Операційні системи

5.4.1 Windows

Windows - найбільш популярна операційна система для персональних комп’ютерів [17]. На цій операційній системи розроблялась і тестувалась частина системи, реалізована на мові програмування Python. Також вузли системи Ethereum можуть виконуватись на цій операційній системі, однак, оскільки система Windows є пропрієтарною, це вимагало б придбання окремої ліцензії для кожного вузла мережі блокчейну. Тому система була розроблена на Windows, але для реальної роботи було створено середовище з віртуальних машин, що працюють під управлінням операційної системи Linux.

5.4.2 Linux

Linux - загальна назва UNIX-подібних операційних систем на основі однойменного ядра. Linux портовано на велику кількість апаратних платформ [18]. Тепер ця ОС досить успішно використовується як на мейнфреймах та суперкомп'ютерах, так і вбудована в багато інших пристроїв (смартфони, планшетні ПК, маршрутизатори комп'ютерних мереж (роутери), пристрої автоматики, системи керування телевізорами та ігровими консолями тощо) [19]. Є багато дистрибутивів цієї операційної системи, що розповсюджуються безкоштовно, працюють стабільно та мають усі необхідні для адміністрування функції. Систему керування режимами електричної мікромережі було протестовано на наступних Linux-дистрибутивах:

- Ubuntu;
- Debian;
- CentOS.

5.5 Системи постійного збереження даних

5.5.1 Sql Server

Microsoft SQL Server — комерційна система керування базами даних, що розповсюджується корпорацією Microsoft. Ця система ефективно використовується як для невеликих і середніх за розміром баз даних, так і для великих баз даних масштабу підприємства. Вона багато років активно та дуже вдало конкурує з іншими системами керування базами даних [20]. Мова, що використовується для запитів — Transact-SQL.

Transact-SQL є реалізацією стандарту ANSI / ISO щодо структурованої мови запитів SQL із розширеннями. Transact-SQL додає до стандартного SQL наступні можливості:

- керуючі оператори,
- локальні і глобальні змінні,
- різні додаткові функції для обробки рядків, дат, математики, тощо,
- підтримка аутентифікації Microsoft Windows.

Оскільки система управління базами даних доступна безкоштовно, вона є кроссплатформенною.

У таблиці 5.4 наведено ключові особливості системи управління базами даних Microsoft SQL Server, що були прийняті до уваги під час вибору СУБД для розробки системи управління режимами електричної мікромережі.

Таблиця 5.4 - Основні характеристики системи управління базами даних Microsoft SQL Server

№	Назва	Опис
1	Здатність витримувати навантаження	Повністю покриває вимоги по навантаженню, системи управління станами електричної мікромережі
2	Доступність	Безкоштовно доступна для некомерційних проектів
3	Підтримка ОС	Windows, RHEL, SUSE, Ubuntu
4	Інструментарій	SQL Server Management Studio

5.5.2 Ethereum

Ethereum - криптовалюта та платформа для створення децентралізованих онлайн-сервісів на базі блокчейна, що працює на основі розумних контрактів. Реалізована як єдина децентралізована віртуальна машина [6].

Ethereum є першою системою, що реалізувала концепцію розумних контрактів на основі блокчейнів. При запуску на блокчейн розумний контракт стає схожим на самостійну комп'ютерну програму, яка автоматично виконується, коли виконуються певні умови. Блокчейн гарантує, що розумні контракти завжди безперебійно будуть працювати без будь-якої можливості простою, цензури, шахрайства або втручання третіх сторін.

У якості сховища для зберігання даних про модель електричної мікромережі було обрано Ethereum блокчейн, а для описання її роботи було реалізовано смарт-контракт, що моделює роботу системи.

5.6 Сервери за стосунків

5.6.1 Nginx

Nginx - вільний веб-сервер і проксі-сервер. Є версії для сімейства Unix-подібних операційних систем (FreeBSD, GNU/Linux, Solaris, Mac OS X) та Microsoft Windows [21]. Основні функції серверу:

- обслуговування статичних запитів, індексних файлів, автоматичне створення списку файлів, кеш дескрипторів відкритих файлів
- акселероване проксіювання з підтримкою кешування;
- акселерована підтримка FastCGI і memcached серверів, простий розподіл навантаження і відмовостійкість;
- модульність, фільтри, gzip, byte-ranges (докачка), chunked відповіді, HTTP-аутентифікація, SSI-фільтр;
- вкладені запити на одній сторінці виконуються паралельно;
- підтримка SSL;
- експериментальна підтримка вбудованого Perl;
- експериментальна підтримка HTTP/2;
- перенаправлення користувача на IMAP/POP3-бекенд з використання зовнішнього HTTP-сервера аутентифікації;
- проста аутентифікація (LOGIN, USER/PASS);
- підтримка SSL і StartTLS [22].

Для обслуговування запитів до системи керування режимами функціонування електричної мікромережі було обрано веб-сервер Nginx через те, що він покриває вимоги системи по навантаженню, є простим у налаштування та розповсюджується безкоштовно.

5.7 Інтегровані середовища розробки

5.7.1 Visual Studio Code

Visual Studio Code — засіб для створення, редагування та налагодження сучасних веб-застосунків і програм для хмарних систем. Visual Studio Code розповсюджується безкоштовно і доступний у версіях для платформ Windows, Linux та OS X. Редактор містить вбудовані інструменти для роботи з Git і засоби рефакторингу, навігації по коду, автодоповнення типових конструкцій і контекстної підказки [23]. Visual Studio Code підтримує розробку на мові програмування Python та розповсюджується безкоштовно, тому частина системи керування режимами електричної мікромережі, що реалізована на мовах Python та Javascript, розроблялась у цьому середовищі на операційній системі Windows.

5.7.2 Remix

Remix - це набір інструментів для взаємодії з блокчейном Ethereum для налагодження транзакцій. Він містить у собі компілятор смарт-контрактів та веб-налагоджувач транзакцій [24].

Remix IDE - це інтегроване середовище розробки для розробників розподілених систем за допомогою мови програмування Solidity, що працює на основі Remix.

Система Remix IDE є безкоштовною і не потребує складної конфігурації середовища для роботи. Інтегроване середовище розробки виконується у вікні браузера.

5.8 Протоколи передачі даних

5.8.1 HTTPS

HTTPS - схема URI, що синтаксично ідентична схемі HTTP, яка зазвичай використовується для доступу до ресурсів у інтернеті.

Використання HTTPS вказує на те, що використовується протокол HTTP з іншим портом за замовчуванням (443) і додатковим шаром шифрування/автентифікації між HTTP і TCP [25].

Оскільки більшість сучасних браузерів відмічають усі веб-ресурси, що віддаються по HTTP URI, як небезпечні, фронтенд-сервер системи керування режимами електричної мікромережі доступний через протокол HTTPS.

У разі, якщо користувач запросить ресурси по HTTP URI, його буде автоматично спрямовано на HTTPS URI.

5.8.2 RPC

RPC (Виклик віддалених процедур) - протокол, що дозволяє програмі, запущеній на одному комп'ютері, звертатись до функцій (процедур) програми, що виконується на іншому комп'ютері, подібно до того, як програма звертається до власних локальних функцій [26].

Це рекомендований до використання спосіб взаємодії з системою блокчейну, що працює на основі технології Ethereum.

5.9 Службові інструменти

5.9.1 Git

Git — розподілена система керування версіями файлів та спільної роботи. Git є однією з найефективніших, надійних і високопродуктивних систем керування версіями, що надає гнучкі засоби нелінійної розробки, що базуються на відгалуженні і злитті гілок. Для забезпечення цілісності історії та стійкості до змін заднім числом використовуються криптографічні методи, також можлива прив'язка цифрових підписів розробників до тегів і комітів. Програма є вільною і випущена під ліцензією GNU GPL версії 2 [27].

Основні можливості, що надає система керування версіями Git, які роблять цю систему оптимальною для розробки системи керування режимами електричної мікромережі:

- збереження файлів;
- локальні операції;
- цілісність даних;
- галуження;
- зливання та перебазовування даних.

5.9.2 Github

GitHub - один з найбільших веб-сервісів для спільної розробки програмного забезпечення. Сервіс безкоштовний для проектів з відкритим вихідним кодом, з наданням користувачам усіх своїх можливостей [28].

Для розробки системи керування режимами електричної мікромережі сервіс Github використовувався у якості віддаленого хостингу Git репозиторію з вихідним кодом.

5.9.3 Github Actions

Actions - сервіс, безкоштовно доступний для відкритих репозиторіїв у сервісі Github, що дозволяє автоматизувати інфраструктурні задачі проекту розробки програмного забезпечення. Під час розробки системи керування режимами електричної мікромережі сервіс Github Actions використовувався для неперервної інтеграції, автоматичного запуску тестів та автоматичного деплойменту. Після будь-якого коміту у сторонню гілку, сервіс автоматично виконує усі тести, а після коміту у головну гілку відбувається автоматичне розгортання системи у робочому середовищі [29].

5.9.4 DigitalOcean

DigitalOcean - це сервіс, що пропонує хмарну інфраструктуру, яка призначена для вільного використання широким загалом [30]. Для розробки системи керування режимами електричної мікромережі було обрано саме цього провайдера хмарної інфраструктури, оскільки він працює достатньо надійно, дає можливість розгортання віртуальних машин, інтегрується з Github Actions та надає студентам можливість пробного безкоштовного періоду.

5.9.5 Trello

Trello — безкоштовна багатоплатформна система управління проектами. Вона використовує парадигму керування проектами, відому як канбан. Проекти

зображуються дошками, що містять списки. Списки містять картки, якими зображуються задачі. Картки повинні переходити з попереднього списку до наступного (за допомогою перетягування), таким чином зображаючи рух якоїсь функції від ідеї, аж до тестування. Картці може бути присвоєно відповідальних за неї користувачів. Користувачі та дошки можуть об'єднуватись в команди [31].

У ході роботи над системою керування режимами електричної мікромережі сервіс Trello використовувався для керування життєвим циклом проекту та планування.

5.10 Висновок розділу

Для розробки бізнес-логіки системи було використано мову програмування Python. Клієнтську частину було реалізовано за допомогою мови програмування Javascript. Смарт-контракт було реалізовано за допомогою мови програмування Solidity. Для розробки системи було використано фреймворки Flask та React та бібліотеки Web3, Gunicorn та Autopep8. Система розроблялась та тестувалась на наступних операційних системах:

- Windows;
- Ubuntu;
- Debian;
- CentOS.

Для постійного збереження даних використовуються технології SqlServer та Ethereum. Веб-запити до системи обслуговує веб-сервер Nginx. Система керування режимами електричної мікромережі була розроблена у інтегрованому середовищі розробки Visual Studio Code та системі Remix IDE. Фронтенд сервер взаємодіє з клієнтом через протокол HTTPS. Система взаємодіє з блокчейном за допомогою

протоколу RPC. Під час роботи над проектом було використано наступні службові інструменти:

- Git;
- Github;
- Github Actions;
- DigitalOcean;
- Trello.

6. ER-ДІАГРАМА

У додатку А наведено ER-діаграму системи.

6.1 Таблиця Users

Таблиця Users зберігає облікові дані усіх користувачів, зареєстрованих у системі. У таблиці 6.1 наведено опис усіх полів таблиці Users.

Таблиця 6.1 - Опис полів таблиці Users

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор користувача у системі.
2	Login	nvarchar (max)	Ні	Ім’я користувача, що використовується під час авторизації у системі.
3	PasswordHash	nvarchar (max)	Ні	Хеш пароля доступу в систему користувача.
4	Role_Id	int	Ні	Роль, яку користувач має у системі.
5	Settings_Id	int	Так	Налаштування системи, збережені користувачем.

У таблиці 6.2 описано усі ключі таблиці Users.

Таблиця 6.2 - Ключі таблиці Users

№	Назва	Тип
1	Id	Primary Key
2	Role_Id	Foreign Key
3	Settings_Id	Foreign Key

6.2 Таблиця UserRole

Таблиця UserRole зберігає всі ролі, які можуть мати користувачі у системі. У таблиці 6.3 наведено опис усіх полів таблиці UserRole.

Таблиця 6.3 - Опис полів таблиці UserRole

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор ролі у системі.
2	Name	nvarchar (max)	Ні	Назва ролі.

У таблиці 6.4 описано усі ключі таблиці UserRole.

Таблиця 6.4 - Ключі таблиці UserRole

№	Назва	Тип
1	Id	Primary Key

Ролі, які існують у системі:

- Observer (Спостерігач);
- Worker (Працівник);
- Admin (Адміністратор).

6.3 Таблиця Settings

Таблиця Settings зберігає персональні налаштування користувачів у системі. Під час автентифікації кожного користувача, налаштування застосовуються для його сесії. У таблиці 6.5 наведено опис усіх полів таблиці Settings.

Таблиця 6.5 - Опис полів таблиці Settings

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор налаштування у системі.
2	Timezone	nvarchar(max)	Так	Часовий пояс, у якому знаходиться користувач.

Продовження таблиці 6.5

3	Language_Id	int	Так	Мова інтерфейсу
4	Theme_Id	int	Так	Тема оформлення інтерфейсу, яку використовує користувач.

У таблиці 6.6 описано усі ключі таблиці Settings.

Таблиця 6.6 - Ключі таблиці Settings

№	Назва	Тип
1	Id	Primary Key

6.4 Таблиця Language

Таблиця Language призначена для збереження мов локалізації інтерфейсу системи. У таблиці 6.7 наведено опис усіх полів таблиці Language.

Таблиця 6.7 - Опис полів таблиці Language

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор мови у системі.
2	Name	nvarchar (max)	Ні	Назва мови.

У таблиці 6.8 описано усі ключі таблиці Language.

Таблиця 6.8 - Ключі таблиці Language

№	Назва	Тип
1	Id	Primary Key

Доступні мови локалізації системи:

- англійська;
- українська;
- російська.

6.5 Таблиця Theme

Таблиця Theme призначена для збереження тем інтерфейсу системи. У таблиці 6.9 наведено опис усіх полів таблиці Language.

Таблиця 6.9 - Опис полів таблиці Theme

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор теми у системі.
2	Name	nvarchar (max)	Ні	Назва теми.

У таблиці 6.10 описано усі ключі таблиці Language.

Таблиця 6.10 - Ключі таблиці Theme

№	Назва	Тип
1	Id	Primary Key

Доступні теми інтерфейсу системи:

- light (світла);
- dark (темна).

6.6 Таблиця GridSnapshots

Таблиця GridSnapshots призначена для збереження станів моделей електричної мікромережі у системі. У таблиці 6.11 наведено опис усіх полів таблиці GridSnapshots.

Таблиця 6.11 - Опис полів таблиці GridSnapshots

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор стану електричної мікромережі у системі.
2	Name	nvarchar (max)	Ні	Назва стану електричної мікромережі.

Продовження таблиці 6.11

№	Назва	Тип	Допускає null	Опис
3	CreationDate	datetime	Ні	Дата і час збереження стану електричної мікромережі
4	User_Id	int	Ні	Користувач, який зберіг стан електричної мікромережі

У таблиці 6.12 описано всі ключі таблиці GridSnapshots.

Таблиця 6.12 - Ключі таблиці GridSnapshots

№	Назва	Тип
1	Id	Primary Key
2	User_Id	Foreign Key

6.7 Таблиця Networks

Таблиця Networks призначена для збереження ланок моделей електричної мікромережі у системі.

У таблиці 6.13 наведено опис усіх полів таблиці Networks.

Таблиця 6.13 - Опис полів таблиці Networks

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор.
2	Name	nvarchar (max)	Ні	Назва ролі.
3	GridSnapshot	int	Ні	Стан електричної мікромережі, якому належить ланка.

У таблиці 6.14 описано всі ключі таблиці Networks.

Таблиця 6.14 - Ключі таблиці Networks

№	Назва	Тип
1	Id	Primary Key
2	GridSnapshot_Id	Foreign Key

6.8 Таблиця ProviderNodes

Таблиця ProviderNodes призначена для збереження вузлів моделей електричної мікромережі, що виробляють електроенергію, у системі.

У таблиці 6.15 наведено опис усіх полів таблиці ProviderNodes.

Таблиця 6.15 - Опис полів таблиці ProviderNodes

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Ідентифікатор вузла.
2	Name	nvarchar (max)	Ні	Назва вузла.
3	Network_Id	int	Ні	Ланка електричної мікромережі, якій належить вузол.
4	ProvideAvg	float	Ні	Середній обсяг вироблення електроенергії
5	ProvideMax	float	Ні	Середній обсяг вироблення електроенергії
4	ProvideCurrent	float	Ні	Середній обсяг вироблення електроенергії
5	NodeType_Id	int	Ні	Тип вузла

У таблиці 6.16 описано всі ключі таблиці ProviderNodes.

Таблиця 6.16 - Ключі таблиці ProviderNodes

№	Назва	Тип
1	Id	Primary Key
2	Network_Id	Foreign Key
3	NodeType_Id	Foreign Key

6.9 Таблиця ConsumerNodes

Таблиця ConsumerNodes призначена для збереження вузлів моделей електричної мікромережі, що виробляють електроенергію, у системі. У таблиці 6.17 наведено опис усіх полів таблиці ConsumerNodes.

Таблиця 6.17 - Опис полів таблиці ConsumerNodes

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор вузла у системі.
2	Name	nvarchar (max)	Ні	Назва вузла.
3	Network_Id	int	Ні	Ланка електричної мікромережі, якій належить вузол.

Продовження таблиці 6.17

№	Назва	Тип	Допускає null	Опис
4	ConsumeAvg	float	Ні	Середній обсяг споживання електроенергії
5	ConsumeMax	float	Ні	Середній обсяг споживання електроенергії
6	ConsumeCurr	float	Ні	Середній обсяг споживання електроенергії
7	NodeType_Id	int	Ні	Тип вузла

У таблиці 6.18 описано всі ключі таблиці ConsumerNodes.

Таблиця 6.18 - Ключі таблиці ConsumerNodes

№	Назва	Тип
1	Id	Primary Key
2	Network_Id	Foreign Key
3	NodeType_Id	Foreign Key

6.10 Таблиця MediatorNodes

Таблиця MediatorNodes призначена для збереження вузлів-посередників електричної мікромережі у системі.

Цей тип вузлів не виробляє і не споживає електричну енергію, а використовується для з’єднання між собою незалежних ланок мережі, якщо це необхідно.

У таблиці 6.19 наведено опис усіх полів таблиці MediatorNodes.

Таблиця 6.19 - Опис полів таблиці MediatorNodes

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор вузла у системі.
2	Name	nvarchar (max)	Ні	Назва вузла.
3	Network_Id	int	Ні	Ланка електричної мікромережі, якій належить вузол.
4	NodeType_Id	int	Ні	Тип вузла

У таблиці 6.20 описано всі ключі таблиці MediatorNodes.

Таблиця 6.20 - Ключі таблиці MediatorNodes

№	Назва	Тип
1	Id	Primary Key
2	Network_Id	Foreign Key
3	NodeType_Id	Foreign Key

6.11 Таблиця NodeType

Таблиця NodeType призначена для збереження типів вузлів електричної мікромережі у системі. У таблиці 6.21 наведено опис усіх полів таблиці NodeType.

Таблиця 6.21 - Опис полів таблиці NodeType

№	Назва	Тип	Допускає null	Опис
1	Id	int	Ні	Унікальний ідентифікатор типу вузла у системі.
2	Name	nvarchar (max)	Ні	Назва типу вузла.

У системі існують наступні типи вузлів:

- атомна електростанція;
- теплова електростанція;
- гідроелектрична станція;
- вітроелектростанція;

- геотермальна електростанція;
- сонячна електростанція;
- будинок;
- виробниче приміщення;
- трансформатор.

6.12 Висновок розділу

Таблиці, що формують ER-модель системи:

- Users
- UserRole
- Settings
- Language
- Theme
- GridSnapshots
- Networks
- ProviderNodes
- ConsumerNodes
- MediatorNodes
- NodeType

7. РЕАЛІЗАЦІЯ БІЗНЕС-ЛОГІКИ СИСТЕМИ

7.1 Архатектура

7.1.1 Мікросервіси

При проектуванні системи було прийнято рішення обрати мікросервісну архітектуру для забезпечення горизонтального масштабування та зручності розгортання.

Термін "Мікросервісна архітектура" з'явився протягом останніх кількох років, щоб описати особливий спосіб проектування програмних застосувань як наборів самостійно розгорнутих сервісів. Хоча точного визначення цього архітектурного стилю не існує, існують певні загальні характеристики організації навколо задовільнення вимог бізнесу, автоматичного розгортання, інтелекту на кінцевих точках та децентралізованого контролю над мовами та даними.

Архітектурний стиль мікросервісів - це підхід до розробки єдиної програми як набору дрібних сервісів, кожен з яких працює у власному процесі та спілкується за допомогою легких механізмів, часто - API ресурсів HTTP. Ці служби будуються на основі бізнес-вимог і незалежно розгортаються за допомогою повністю автоматизованого механізму розгортання. Існує мінімальне централізоване управління цими службами, які можуть бути написані на різних мовах програмування та використовувати різні технології зберігання даних [32].

У системі керування режимами електричною мікромережею було реалізовано наступні мікросервіси:

- фронтенд-сервіс
- сервіс керування моделлю
- сервіс взаємодії з блокчейном

- сервіс взаємодії з базою даних
- сервіс взаємодії з блокчейном
- блокчейн.

7.1.2 MVC

Фронтенд-сервіс системи реалізовано на основі архітектурного патерну MVC. Даний патерн передбачає поділ системи на три взаємопов’язані компоненти:

- модель даних;
- вигляд (інтерфейс користувача);
- модуль керування [33].

У реалізації фронтенд-сервісу системи управління режимами електричної мікромережі було застосовано даний патерн для відокремлення даних від інтерфейсу користувача так, щоб зміни інтерфейсу користувача мінімально впливали на роботу з даними, а зміни в моделі даних могли здійснюватися без змін інтерфейсу користувача.

7.1.3 CQRS

Command query responsibility segregation - принцип імперативного програмування, що полягає у логічному розділенні операцій доменної області на дві категорії:

- запити;
- команди.

Запити можуть читати дані з постійного сховища, чи кешу, не змінюючи при цьому стан системи. Команди спрямовані на передачу певних повідомлень системі з метою зміни її стану та ініціювання певної поведінки [34].

Під час розробки системи керування режимами електричної мікромережі підхід CQRS було використано з метою спрощення об’єктів доменної області. Цього вдалось досягнути через те, що, внаслідок розділення операцій системи на запити та команди, логіка запису, виклику поведінки та читання була рознесена по окремим модулям, що дозволило оперувати більш простими об’єктами, що притримуються принципу Single Responsibility.

7.1.4 Event Sourcing

Принцип даного підходу полягає у збереженні кожної події, що змінює стан системи. Таким чином використання підходу Event Sourcing дозволяє відновити стан системи у будь-який момент її функціонування. У якості постійного сховища доменних подій системи виступає система Ethereum.

7.1.5 SOLID

SOLID - набір принципів об’єктно-орієнтованого проектування та дизайну, покликаний сприяти розробці програмних систем, здатних то змін, розширення та підтримки [35].

У таблиці 7.1 наведено опис кожного з принципів SOLID.

Таблиця 7.1 - SOLID принципи

№	Назва	Опис
1	Single responsibility principle	Кожен об'єкт має виконувати лише один обов'язок.

Продовження таблиці 7.1

№	Назва	Опис
2	Open/closed principle	Програмні сутності повинні бути відкритими для розширення, але закритими для змін.
3	Liskov substitution principle	Об'єкти в програмі можуть бути замінені їх нащадками без зміни коду програми.
4	Interface segregation principle	Багато спеціалізованих інтерфейсів краще за один універсальний.
5	Dependency inversion principle	Абстракції не повинні залежати від деталей реалізації.

Використання принципів SOLID дозволило побудувати систему, більш здатну до змін та підтримки. Реалізована система керування режимами електричної мікромережі складається з самодостатніх модулів, слабо зв’язаних між собою, що дозволяє повторно використовувати їх у разі необхідності.

7.2 Патерни

7.2.1 Abstract Factory

Абстрактна фабрика — це породжуючий патерн проектування, який вирішує проблему створення сімейств пов’язаних об’єктів, без прив’язки коду до конкретних класів продуктів [36]. На рисунку 7.1 зображено структурну схему патерна Абстрактна фабрика.

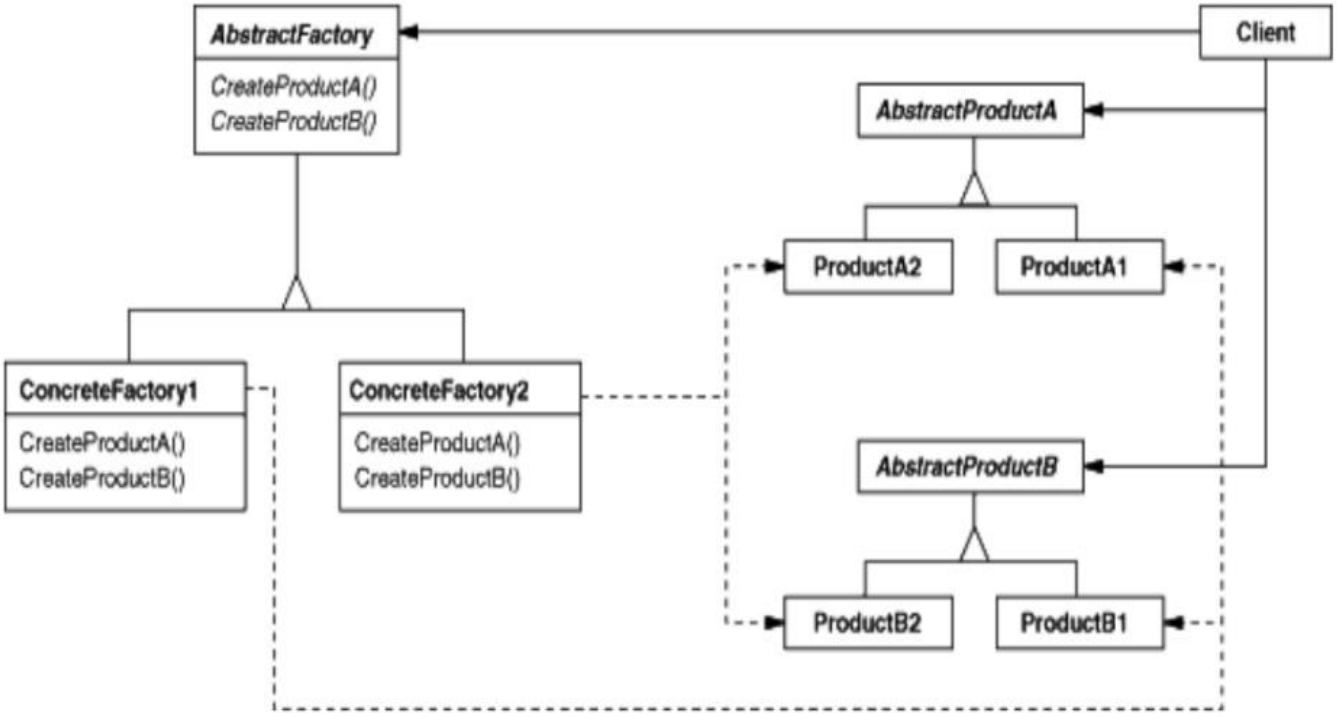

Рисунок 7.1 - Структурна схема патерна Абстрактна фабрика [36]

Оскільки електрична мікромережа є кореневою сутністю обмеженого контексту, вона складається з великої кількості пов’язаних між собою об’єктів, таких як ланки мережі, які у свою чергу складаються з вузлів. Абстрагування типів об’єктів під час створення їх сімейства дозволяє легко вводити нові типи, не змінюючи існуючого коду. Це дозволяє притримуватися принципу Open/Closed. На лістингу 1 зображено спрощений процес створення моделі електричної мікромережі за допомогою патерна Абстрактна фабрика. У даному лістингу ігнорується використання патерна Будівельник, який розбиває побудову різних компонентів електричної мікромережі на окремі логічні кроки.

7.2.2 Builder

Будівельник - це породжувальний патерн проектування, що дає змогу створювати складні об’єкти крок за кроком. Він дає можливість використовувати один і той самий код будівництва для отримання різних відображень об’єктів [36].

Модель електричної мікромережі є порівняно складною системою, що складається з великої кількості об’єктів - ланок та вузлів. Побудову моделі мікромережі можна декомпозувати на три окремі логічні кроки:

- побудувати вузли
- побудувати ланки
- побудувати мережу

Це робить доцільним використання патерна Будівельник для побудови об’єктної моделі електричної мікромережі.

На рисунку 7.2 зображено структурну схему патерна Будівельник.

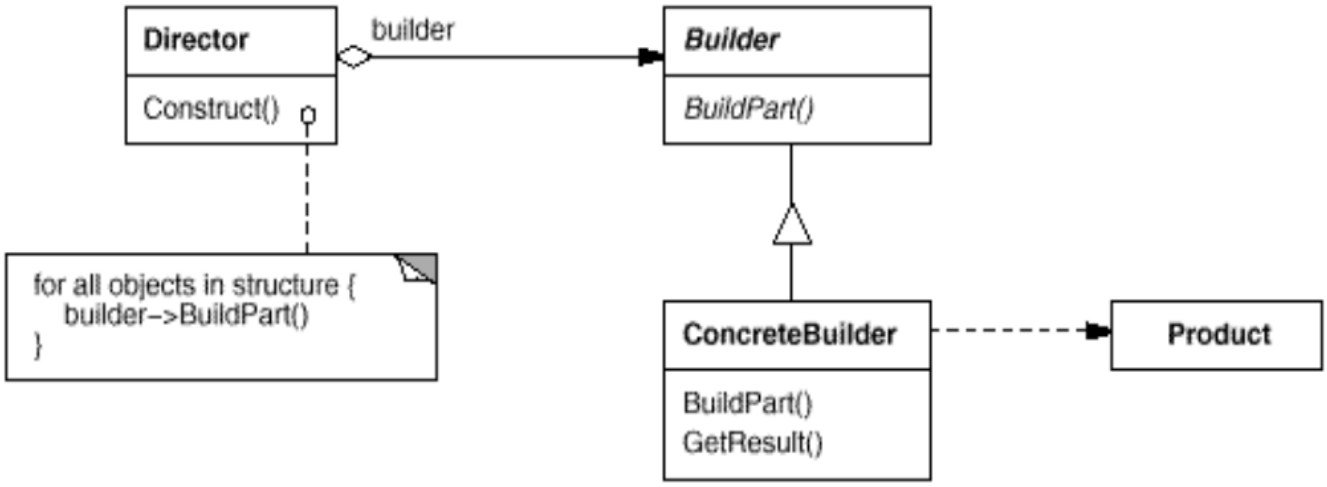

Рисунок 7.2 - Структурна схема патерна Будівельник [36]

7.2.3 Facade

Фасад — це структурний патерн проектування, який надає простий інтерфейс до складної системи класів, бібліотеки або фреймворку [36]. На рисунку 7.3 зображено структурну схему патерна Фасад.

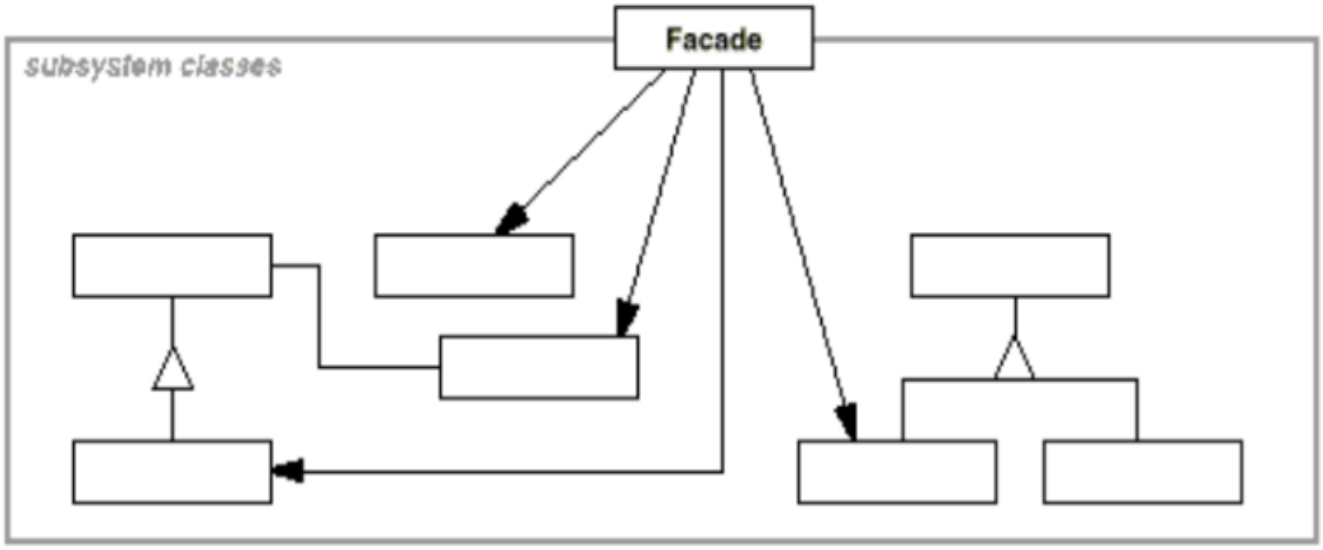

Рисунок 7.3 - Структурна схема патерна Фасад [36]

У системі керування режимами електричної мікромережі патерн Фасад використовується у тих місцях, де необхідно обмежити складні зовнішні інтерфейси лише тими викликами, які потрібні системі та зробити роботу з ними максимально простою та прозорою. Тому даний патерн будо реалізовано у наступних модулях системи:

- модуль взаємодії з блокчейном;
- модуль взаємодії з базою даних.

7.2.4 Composite

Компонувальник — це структурний патерн проектування, що дає змогу згрупувати декілька об’єктів у деревоподібну структуру, а потім працювати з нею так, ніби це одиничний об’єкт [366].

Реалізація патерна Компонувальник дозволяє легко модифікувати складні доменні об’єкти системи:

- мережі;
- ланки.

На рисунку 7.4 зображено структурну схему патерна Компонувальник.

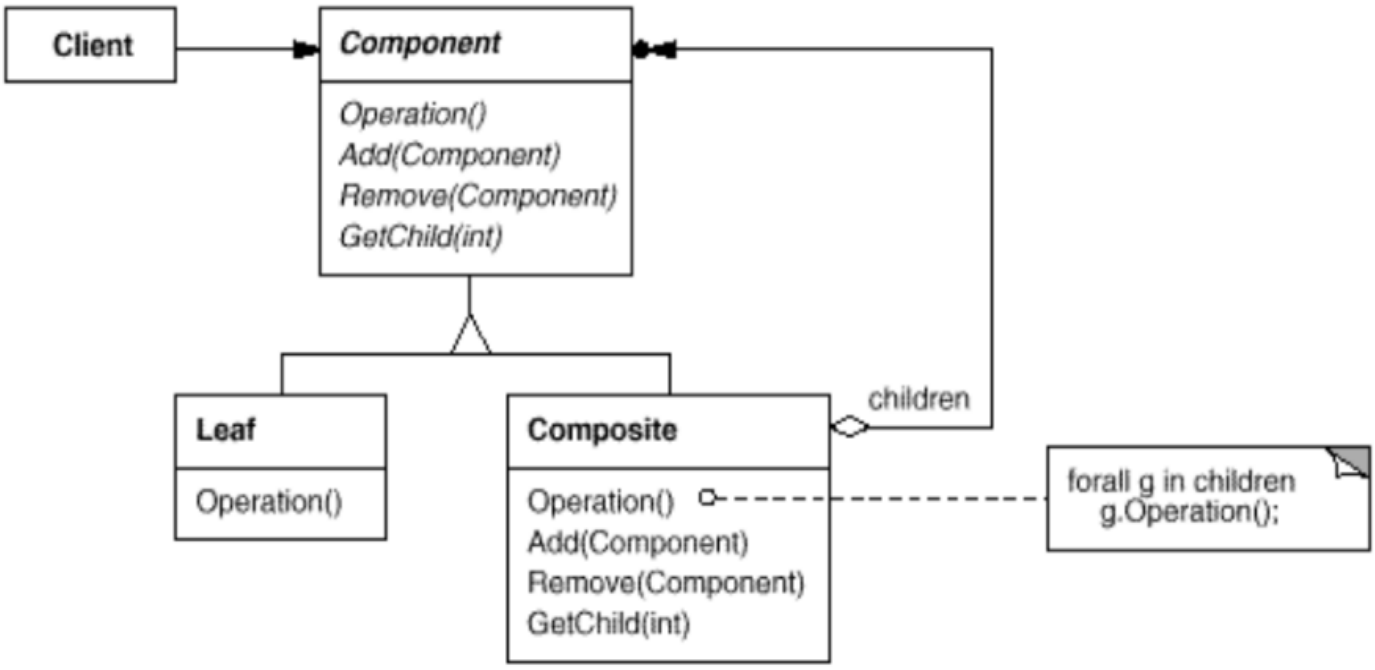

Рисунок 7.4 - Структурна схема патерна Компонувальник [36]

7.3 Висновок розділу

Під час розробки архітектури системи керування режимами електричної мікромережі було використано наступні архітектурні патерни та підходи:

- мікросервіси;
- MVC;
- CQRS;
- Event Sourcing;
- SOLID.

Також під час розробки дизайну системи було використано наступні патерни об’єктно-орієнтованого проектування:

- Abstract Factory;
- Builder;
- Façade;
- Composite.

8. РОЗРОБЛЕННЯ ІНТЕРФЕЙСА КОРИСТУВАЧА

8.1 Побудова інтерфейса у браузері

Систему керування режимами електричної мікромережі реалізовано таким чином, що клієнт взаємодіє з її серверною частиною через веб-клієнт, що виконується в контексті браузера. При проектуванні системи було обрано саме цей підхід через те, що він дає наступні переваги:

- мультиплатформенність за замовчуванням. Додаток інтерпретується веб-браузером і не залежить від оточення операційної системи, тому немає необхідності у платформо-залежному коді;
- додаток не потребує встановлення. Усі ресурси, необхідні для роботи додатку, повертаються у відповідь на запит до сервера;
- програма працює у ізольованому середовищі браузера, що робить її більш безпечною;
- тонкий клієнт у браузері дозволяє виконувати усі важкі обчислення на сервері, тому системні вимоги до апаратного забезпечення користувача можуть бути досить малими.

Всі сучасні браузери підтримують мову розмітки гіпертекстових документів (HTML), каскадні таблиці стилів (CSS) та динамічну мову створення сценаріїв (Javascript), що використовується для реалізації інтерактивної взаємодії користувача з системою. Тому під час розробки клієнтської частини системи керування режимами електричної мікромережі було використано ці три основні технології веб-розробки та надбудови над ними. Для організації Javascript-коду у більш масштабованому та здатному до підтримки вигляді було використано фреймворк React, а з метою перевикористання загальновживаних компонентів стилів було використано бібліотеку

Bootstrap. Для спрощення рутинних операцій та гарантування підтримки усіх сучасних браузерів було використано бібліотеку JQuery.

8.2 React

Основна логіка роботи системи організована за принципами, що пропонує фреймворк React.

У функції ReactDOM.render описується кореневий вузол додатку, який є контейнером для усіх інших. Так само у кожен з вузлів можуть вкладатися інші вузли DOM-об’єкту.

Таким чином кожен фрагмент сторінки, що є логічно окремим, або потребує окремого режиму оновлення, чи логіки, описується, як окремий компонент. Такий підхід дає змогу більш гранулярного розбиття користувацьких елементів управління, що дозволяє легше вносити правки у систему, розширювати та змінювати її. Концепт Virtual DOM дозволяє реактивно змінювати сторінку на основі зміни властивостей Javascript - об’єктів.

8.3 Virtual DOM

Віртуальний DOM - це концепція програмування інтерфейсів у браузері, де віртуальне представлення інтерфейсу зберігається в пам'яті та синхронізується з реальним DOM-об’єктом браузера за допомогою бібліотеки ReactDOM. Цей процес називається reconciliation. Цей підхід дозволяє використовувати декларативний інтерфейс фреймворка React. Модуль, що використовує даний інтерфейс, повідомляє React, у якому стані інтерфейс користувача повинен знаходитися, і React гарантує, що DOM відповідає цьому стану. Це абстрагує маніпуляції з атрибутами, обробку подій та оновлення DOM вручну, які інакше доведеться використовувати для створення

програми. Це значно спрощує розробку інтерфейсу у браузері, оскільки програма абстрагується від реального DOM-об’єкта браузера. Програма керує своїми об’єктами, змінюючи їх властивості. А фреймворк слідкує за тим, щоб кожна необхідна зміна відповідних об’єктів була відображена не DOM-об’єкті вікна.

8.4 Bootstrap

Bootstrap - це набір інструментів для розробки за допомогою HTML, CSS та JavaScript. Він дозволяє швидко прототипувати продукт та будувати веб-додатки за допомогою адаптивної системи сітки, великої кількості попередньо вбудованих компонентів та потужних плагінів, побудованих на jQuery. Bootstrap підтримує адаптивну верстку, підтримується усіми сучасними браузерами та надає можливість повторного використання складних елементів користувацького інтерфейсу, таких як інтерактивні таблиці, сітки даних та інструменти вводу даних. Також дана бібліотека дає широкі можливості стилізації даних компонентів.

8.5 JQuery

JQuery - це швидка, невелика та багатофункціональна бібліотека JavaScript. Завдяки простому у користуванні API, який працює в усіх сучасних браузерах, такі речі, як обробка та маніпулювання документами HTML, обробка подій, анімація та Ajax, стають набагато простішими. JQuery підтримує інтеграцію з іншими технологіями, що використовуються під час реалізації клієнтської частини системи, такими як React та Bootstrap.

8.6 Висновок розділу

Клієнтська частина системи керування режимами електричної мікромережі реалізована у вигляді веб-додатку, що виконується у браузері. Додаток підтримує всі сучасні браузери.

Технології, що були використані при розробці клієнтської частини системи керування режимами електричної мікромережі:

- HTML;
- CSS;
- Javascript;
- React;
- Bootstrap;
- JQuery;
- react-vis.

9. ТЕСТУВАННЯ СИСТЕМИ

9.1 Фреймворк для тестування

В ході написання тестів для серверної частини системи керування режимами електричної мікромережі використовувався вбудований в мову Python інструмент - unittest. Це фреймворк для тестування, що має програмний інтерфейс, схожий на JUnit.

Основні особливості фреймворку Python unittest:

- підтримує автоматичний запуск тестів;
- дозволяє повторно використовувати логіку налаштування середовища та звільнення ресурсів для багатьох тестів;
- дозволяє агрегувати тести у логічно зв’язані колекції;
- працює незалежно від фреймворку для формування звітів;

Основні концепцій фреймворка unittest:

- Test fixture - логіка підготовки, необхідної для роботи одного або декількох тестів, та будь-які пов'язані з цим дії звільнення ресурсів;
- Test case - індивідуальна одиниця тестування. Він перевіряє конкретну реакцію системи на певний набір вхідних даних. unittest пропонує базовий клас TestCase, який може бути використаний для створення нових тестових випадків;
- Test suite - це сукупність тестових випадків та/або тестових наборів. Він використовується для логічного об'єднання тестів, які повинні виконуватися разом;
- Test runner - це компонент, який оркеструє виконання тестів і надає користувачеві результат. Він може використовувати графічний інтерфейс, текстовий інтерфейс або повернути спеціальне значення для подальшої візуалізації результатів виконання тестів.

9.2 Dummy-об’єкти

Dummy-об’єкти є найбільш примітивним прикладом тестових об’єктів-двійників. Вони використовуються у випадках, коли для створення об’єкта модуля, що тестується, або для виклику метода, що є частиною тест-кейсу потрібен аргумент, стан та поведінка якого не є суттєвою для даного тест-кейсу. У цьому випадку ця залежність задовольняється “порожнім” об’єктом, виклики функцій якого ні до чого не призводять.

Під час розробки системи керування режимами електричної мікромережі було використано dummy-об’єкти для тестування можливості додавати вузли до існуючої ланки мережі. У цьому тест-кейсі не є важливим, який стан та поведінку має об’єкт вузла, оскільки тестується логіка ланки.

9.3 Fake-об’єкти

Fake-об’єкти використовуються для того, щоб відстежити сторонній ефект модуля та перевірити його коректність.

Під час розробки системи керування режимами електричної мікромережі було використано fake-об’єкти для тестування коректності логіки збереження даних у системі блокчейн. Об’єкт модуля запису даних у систему блокчейн було замінено спеціальним fake-об’єктом, який дозволяє відстежити усі записи до блокчейну.

Замість того, щоб записувати об’єкт у блокчейн, fake-об’єкт зберігає його у списку. Таким чином можна легко валідувати, чи правильний об’єкт було записано до блокчейну.

9.4 Stub-об’єкти

Stub-об’єкти використовуються для того, щоб у ході роботи тест-кейсу об’єкт міг містити у собі всі дані, необхідні для коректної роботи системи, замість того, щоб брати їх із зовнішніх джерел.

Під час розробки системи керування режимами електричної мікромережі було використано stub-об’єкти для того, щоб симулювати отримання даних з блокчейну.

9.5 Spy-об’єкти

Stub-об’єкти використовуються для того, щоб відстежувати непрямі виклики, що робить модуль, робота якого тестується.

Під час розробки системи керування режимами електричної мікромережі було використано spy-об’єкти для того, щоб перевірити, скільки разів викликається логіка збереження даних у блокчейні під час збереження мікромережі, що складається з багатьох елементів.

9.6 Mock-об’єкти

Mock-об’єкти дозволяють зробити попередні налаштування, що описують очікувану поведінку системи, що тестується, а після завершення роботи тест-кейсу - верифікувати, чи усі умови були виконані.

Під час розробки системи керування режимами електричної мікромережі було використано mock-об’єкти для того, щоб перевірити цілісність даних, що надходять до модуля збереження даних у системі блокчейн.

9.7 Покриття коду тестами

Для оцінки покриття коду тестами було використано бібліотеку Coverage.py. Вона контролює виконання тестів, зберігаючи дані про те, які частини коду були виконані, а потім аналізує вихідний код, щоб покриті и не покриті тестами ділянки. Кодова база проекту покрита тестами на 88 відсотків.

9.8 Тест-кейси

9.8.1 Авторизація

У таблиці 9.1 наведено опис тест-кейсу авторизація з коректним ім’ям користувача та паролем.

Таблиця 9.1 - Тест-кейс авторизації з коректним ім’ям користувача та паролем

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку авторизації, вводить логін та пароль та натискає кнопку увійти.	Логін: admin; Пароль: smartgrid	Авторизація успішна.	Так

У таблиці 9.2 наведено опис тест-кейсу авторизація з некоректним ім’ям користувача та паролем.

Таблиця 9.2 - Тест-кейс авторизації з некоректним ім’ям користувача та паролем

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку авторизації, вводить логін та пароль та натискає кнопку увійти.	Логін: wrong; Пароль: wrong	Авторизація невдала.	Так

9.8.2 Створення моделі електричної мікромережі

У таблиці 9.3 наведено опис тест-кейсу створення моделі електричної мікромережі користувачем, що має роль admin.

Таблиця 9.3 - Тест-кейс створення моделі електричної мікромережі користувачем, що має роль admin

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач admin переходить на сторінку створення нової моделі та вводить її ім’я.	Роль: admin Ім’я: SG1	Створення успішне, дані збережено.	Так

У таблиці 9.4 наведено опис тест-кейсу створення моделі електричної мікромережі користувачем, що має роль worker.

Таблиця 9.4 - Тест-кейс створення моделі електричної мікромережі користувачем, що має роль worker

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач admin переходить на сторінку створення нової моделі та вводить її ім’я.	Роль: worker Ім’я: SG1	Користувач не має доступу	Так

9.8.3 Створення ланки електричної мікромережі

У таблиці 9.5 наведено опис тест-кейсу створення ланки електричної мікромережі.

Таблиця 9.5 - Тест-кейс створення ланки електричної мікромережі

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку створення нової ланки для моделі та вводить її ім’я.	Ім’я: Net1	Створення успішне, дані збережено.	Так

9.8.4 Створення вузла у ланці

У таблиці 9.6 наведено опис тест-кейсу створення вузла у ланці електричної мікромережі.

Таблиця 9.6 - Тест-кейс створення вузла у ланці електричної мікромережі

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку створення нового вузла у ланці та вводить його дані.	Ім’я: Net1; Тип: Consumer; ConMax: 25 ConCurr: 12	Створення успішне, дані збережено.	Так

9.8.5 Зміна режиму функціонування вузла

У таблиці 9.7 наведено опис тест-кейсу зміни режиму функціонування вузла у ланці електричної мікромережі.

Таблиця 9.7 - Тест-кейс зміни режиму функціонування вузла у ланці електричної мікромережі

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку створення нового вузла у ланці та вводить його дані.	Ім’я: Net1; Тип: Consumer; ConMax: 25 ConCurr: 12	Створення успішне, дані збережено.	Так

9.8.6 Збереження моделі у базі даних

У таблиці 9.8 наведено опис тест-кейсу збереження поточного стану моделі електричної мікромережі у базі даних.

Таблиця 9.8 - Тест-кейс зміни режиму функціонування вузла у ланці електричної мікромережі

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку збереження стану моделі та натискає кнопку “Зберегти”	Ім’я моделі: SG1	Стан моделі збережено.	Так

9.8.7 Імпорт моделі з бази даних

У таблиці 9.9 наведено опис тест-кейсу імпорту моделі електричної мікромережі з бази даних.

Таблиця 9.9 - Тест-кейс зміни режиму функціонування вузла у ланці електричної мікромережі

Опис тест-кейсу	Вхідні дані	Очікуваний результат	Співпадає з отриманим?
Користувач переходить на сторінку імпорту моделі та натискає кнопку “Імпорт”	Ім’я моделі: SG1	Успішний імпорт	Так

9.9 Висновок розділу

В ході написання тестів для серверної частини системи керування режимами електричної мікромережі використовувався вбудований в мову Python інструмент - unittest. Були використані наступні тестові об’єкти-двійники:

- dummy;
- fake;
- stub;
- spy;
- mock.

Під час тестування системи було розглянуто наступні тест-кейси:

- авторизація
- створення моделі електричної мікромережі
- створення ланки електричної мікромережі
- створення вузла у ланці
- зміна режиму функціонування вузла
- збереження моделі у базі даних
- імпорт моделі з бази даних

Для оцінки покриття коду тестами було використано бібліотеку Coverage.py. Загальне покриття серверної частини системи керування режимами електричної мікромережі - 88%.

10. СТАРТАП-ПРОЕКТ

10.1 Опис ідеї проекту

У таблиці 10.1 наведено зміст ідеї, можливі напрямки застосування та основні вигоди, що може отримати користувач товару за кожним напрямком застосування.

Таблиця 10.1 - Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
	Будівництво селищ з децентралізованим електропостачанням.	Використання даної системи дозволяє досліджувати модель електричної мережі селища у різних режимах до її реальної імплементації.
	Будівництво децентралізованих електричних мереж промислових підприємств.	Використання даної системи дозволяє досліджувати особливості роботи моделі електричної мережі промислового підприємства в умовах різних навантажень до її реальної імплементації.

У таблиці 10.2 наведено опис сильних, слабких та нейтральних характеристик ідеї проекту.

Таблиця 10.2 - Сильні, слабкі та нейтральні характеристики ідеї

№	Техніко-економічні характеристики ідеї	(потенційні) товари/концепції конкурентів			W	N	S
		Мій проект	Energy Plus	Open DDS			
1	Тип сховища даних	Блокчейн	Локальний файл ресурсів	Локальний файл ресурсів	-	-	+
2	Можливість віддаленого доступу	Є	Немає	Немає	-	-	+
3	Візуалізація даних	Немає	Є	Є	+	-	-

10.2 Технологічний аудит ідеї проекту

У таблиці 10.3 наведено дані про технологічну здійсненність проекту.

Таблиця 10.3 - Технологічна здійсненність проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Збереження даних про стан моделі мережі у блокчейні.	Платформа для створення децентралізованих онлайн-сервісів на базі блокчейна.	Існує реалізація, що задовольняє вимоги проекту - Ethereum.	Розповсюджується безкоштовно, вихідних код відкритий.
2	Постійне збереження стану моделі.	Реляційна СУБД.	Існує реалізація, що задовольняє вимоги проекту - SQL Server.	Пропрієтарний продукт, розповсюджується безкоштовно.
3	Веб-інтерфейс	Веб-сервер.	Існує реалізація, що задовольняє вимоги проекту - Nginx.	Розповсюджується безкоштовно, вихідних код відкритий.
Обрана технологія реалізації ідеї проекту: Ethereum + SQL Server + Nginx				

10.3 Аналіз ринкових можливостей запуску стартап-проекту

У таблиці 10.4 наведено дані про аналіз попиту на цільовому ринку.

У результаті аналізу даної таблиці прийнято рішення про те, що за попереднім оцінюванням даний ринок є привабливим для входження, оскільки він швидко росте і не має великої кількості головних гравців.

Таблиця 10.4 - Аналіз попиту на цільовому ринку

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	2
2	Загальний обсяг продаж, грн	50000
3	Динаміка ринку	Зростає
4	Наявність обмежень для входу	Експертиза
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	20

У таблиці 10.5 наведено дані про потенційні групи клієнтів.

Таблиця 10.5 - Потенційні групи клієнтів

№	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Необхідність моделювання особливостей роботи електричних мереж у різних режимах та умовах.	Будівельні та енергетичні компанії.	Будівельні компанії можуть мати нерегулярний попит, оскільки вони не завжди мають проекти будівництва енергетично розподілених селищ.	Стабільність роботи, здатність до масштабування та адаптації до специфічних умов.

У таблиці 10.6 наведено фактори загроз.

Таблиця 10.6 - Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Нестабільність предметної області.	Теорія розумних електромереж бурхливо розвивається та постійно змінюється.	Розробка продукту, який у разі необхідності можно легко змінювати.

Продовження таблиці 10.6

№	Фактор	Зміст загрози	Можлива реакція компанії
2	Збільшення конкуренції на ринку.	Оскільки ринок швидко зростає і зараз на ньому мало гравців, є ймовірність виходу на нього нових компаній.	За рахунок досвіду та експертизи створення продукту найвищої якості, здатного до масштабування та адаптації до різноманітних умов роботи.

У таблиці 10.7 наведено фактори можливостей.

Таблиця 10.7 - Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Ріст ринку	Ринок розумних електричних мереж швидко росте, все більше компаній демонструють зацікавленість.	Активна робота по просуванню свого продукту. Забезпечення найкращого користувацького досвіду та підтримки клієнтів. Гнучке формування цін.
2	Відкритість основних гравців	Основні гравці на ринку тримають вихідний код своїх рішень та специфікації у вільному доступі	Участь у розробці та уніфікацій стандартів, аналіз коду та перейняття найкращих практик.

У таблиці 10.8 наведено ступеневий аналіз конкуренції на ринку.

Таблиця 10.8 - Ступеневий аналіз конкуренції на ринку.

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Чиста	Жоден з гравців не може впливати на загальну ситуацію на ринку.	Інтенсивна робота з користувачами, просування продукту, участь у формуванні відкритих стандартів, забезпечення якості продукту.
Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Глобальна	На ринку присутні гравці з різних країн, що просувають свій продукт користувачам також з різних країн.	Диференціація по регіонах, відкриття регіональних представництв.
Внутрішньогалузева	Усі гравці на ринку діють в одній галузі економіки, виробляють і реалізують однакові товари, що задовольняють одну й ту саму потребу.	Зосередження сил на ринку розробки рішень для будівельних та енергетичних компаній.

Продовження таблиці 10.8

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства
Товарно-родова	Усі гравці на ринку пропонують продукти, що виконують схожі функції.	Зосередження сил на одному продукті, що охоплює всі потреби клієнта.
Нецінова	Користувачі готові платити більше за більш якісний та надійний продукт.	Залучення найбільш компетентних кадрів, застосування ефективних методологій виробництва, виділення ресурсів на тестування.

У таблиці 10.9 наведено аналіз конкуренції в галузі за М. Портером

Таблиця 10.9 - Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
	Energy Plus, Open DDS	Потенційні конкуренти невідомі	Є багато постачальників публічних хмарних ресурсів	Визначити фактори сили споживачів	Фактори загроз з боку замінників
Висновки:	Прямих конкурентів не дуже багато, конкуренція не є інтенсивною.	Наразі жодна компанія не декларує намірів виходити на даний ринок.	Необхідно мінімізувати обчислювальне навантаження на інфраструктуру.	Вимоги споживачів до якості досить високі.	Більш швидка адаптація до вимог ринку.

У таблиці 10.10 наведено обґрунтування факторів конкурентоспроможності.

Таблиця 10.10 - Обґрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Здатність адаптуватися до змін вимог	Технології роботи з інтелектуальними електричними мережами швидко розвиваються та постійно змінюються, тому продукт повинен мати можливість швидко адаптуватися до цих змін.
2	Гарантування якості	Очікування користувачів від якості продукту досить високі, для задоволення їх потреб необхідне програмне забезпечення, що стабільно працює та здатне до масштабування.
3	Підтримка користувача	Користувачам необхідна підтримка, оскільки вони хочуть бути впевнені у тому, що, у разі виникнення неочікуваних подій, вони не втратять результати своєї роботи.
4	Просування на ринку	На ринку розробки програмного забезпечення для моделювання роботи електричних мереж є досить великий попит на рішення для роботи зі Smart Grid. Оскільки лише дві компанії надають такі послуги, просування на ринку може стати сильним фактором конкурентоспроможності.

Таблиця 10.11 містить SWOT-аналіз стартап-проекту

Таблиця 10.11 SWOT-аналіз стартап-проекту

Сильні сторони: централізоване сховище даних, можливість віддаленого доступу.	Слабкі сторони: відсутність візуалізації даних.
Можливості: ріст ринку, відкритість основних гравців.	Загрози: нестабільність предметної області, збільшення конкуренції на ринку.

У таблиці 10.12 наведено альтернативний шлях ринкового впровадження стартап-проекту.

Таблиця 10.12 - Альтернативний шлях ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Розробка додаткового програмного шару візуалізації моделі на основі отриманих від споживачів вимог.	Висока, оскільки користувачі зацікавлені в отриманні продуктів максимальної якості.	6 місяців.

10.4 Розроблення ринкової стратегії проекту

У таблиці 10.13 наведено вибір цільових груп потенційних споживачів.

Таблиця 10.13 - Вибір цільових груп потенційних споживачі

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Складність входу у сегмент
1	Енергетичні компанії	Висока	Високий	Слабка	Низька
2	Будівельні компанії	Висока	Високий	Слабка	Низька
Які цільові групи обрано: енергетичні компанії, будівельні компанії.					

У таблиці 10.14 наведено визначення базової стратегії розвитку.

Таблиця 10.14 - Визначення базової стратегії розвитку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Енергетичні компанії	Забезпечення найкращої на ринку якості та активне просування.	Експертиза, підтримка.	Стратегія спеціалізації
2	Будівельні компанії	Забезпечення найкращої на ринку якості та активне просування.	Експертиза, підтримка.	Стратегія спеціалізації

У таблиці 10.15 наведено визначення базової стратегії конкурентної поведінки

Таблиця 10.15 - Визначення базової стратегії конкурентної поведінки

№	Чи є проект «першопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні.	Оскільки ринок швидко росте, шукати нових та забирати існуючих.	Підходи до вирішення типових задач програмування.	Стратегія лідера.

У таблиці 10.16 наведено визначення базової стратегії позиціонування.

Таблиця 10.16 - Базова стратегія позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Здатність продукту швидко адаптуватися до нових вимог.	Урахування ймовірних факторів змін під час розробки вихідного коду.	Слабка зв’язаність між класами, застосування підходу Open/Closed.	Гнучкість, Орієнтованість на користувача, персоналізація.

Продовження таблиці 10.16

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
2	Високий рівень підтримки.	Виділення ресурсів на інженерів з підтримки.	Можливість ескалувати проблему до інженера-розробника	Відкритість, прозорість, допомога.
3	Високий рівень якості ПЗ.	Додаткове тестування системи.	Використання довгих пауз у зміні вихідного коду в період тестування.	Якість, надійність, відмовостійкість.

10.5 Розроблення маркетингової програми стартап-проекту

У таблиці 10.17 наведено визначення ключових переваг концепції потенційного товару.

Таблиця 10.17 - Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити
1	Можливість використовувати систему, що швидко змінюється у відповідності до вимог бізнесу.	Можливість додавати нову функціональність шляхом розширення існуючої кодової бази, а не її модифікації дозволяє більш швидко змінювати систему.	Більша гнучкість та адаптивність.
2	Можливість звернутися у службу підтримки у разі неочікуваної поведінки системи.	У разі неможливості вирішення проблеми шляхом зміни конфігурації, до роботи над проблемою користувача залучається інженер-розробник.	Можливість швидкого виправлення помилок у конфігурації та вихідному коді.

У таблиці 10.18 наведено опис трьох рівнів моделі товару.

Таблиця 10.18 - Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
I. Товар за задумом	Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди
	Якість: проходження усіх модульних та інтеграційних тестів, мануальне тестування.
	Пакування: відсутнє.
	Марка: Smart Grid Modeling Solution
	Пробний період
	Постійна підтримка
За рахунок чого потенційний товар буде захищено від копіювання: якість, адаптивність, підтримка.	

У таблиці 10.19 наведено визначення меж встановлення ціни.

Таблиця 10.19 - Визначення меж встановлення ціни

№	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	12000	12000	5500000	0-25000

У таблиці 10.20 наведено формування системи збуту.

Таблиця 10.20 - Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Оптимальна система збуту
1	Закупівлі залежать від наявних у клієнтів проектів.	Надання доступу до бінарних виконуваних файлів.	Онлайн-продаж.

У таблиці 10.21 наведено концепцію маркетингових комунікацій.

Таблиця 10.21 - Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Нерегулярність попиту.	Інтернет.	Адаптивність, надійність, підтримка.	Просування товару серед потенційних клієнтів.	Розставлення акцентів на сильных сторонах системи та її доступності.

10.6 Висновок розділу

В умовах ринку розробки рішень моделювання електричних мікромереж наявний попит на подібні рішення, ринок зростає, конкуренція досить низька. Тому є можливість ринкової комерціалізації проекту. Основні групи клієнтів - будівельні та енергетичні компанії. Подальша імплементація проекту є доцільною.

ВИСНОВКИ

В магістерській дисертації було проаналізовано існуючі рішення для моделювання роботи електричних мереж та розроблено мікросервісну архітектуру розподіленої системи управління режимами функціонування електричної мікромережі на основі технології блокчейн. При розробці архітектури було враховано можливість атомарного вертикального та горизонтального масштабування системи та здатність системи до розширення та зміни функціоналу.

У разі збільшення навантаження на систему, для необхідних модулів виділяються додаткові обчислювальні ресурси.

Було реалізовано систему керування режимами функціонування електричної мікромережі та інтегровано її з технологією блокчейн для забезпечення збереження даних моделі. Під час розробки системи було використано підходи SOLID та патерни об’єктно-орієнтованого проектування.

Серверна частина системи розроблялась за допомогою мови програмування Python, мережевий інтерфейс було реалізовано з використанням бібліотеки Flask. Запити користувачів до системи обслуговує веб-сервер Nginx. Додаток взаємодіє з веб-сервером за допомогою WSGI-сервера Gunicorn. У якості блокчейн-платформи для розробки децентралізованих систем було використано Ethereum. Серверна частина системи працює під управлінням операційних систем Windows та Linux.

Клієнтська частина системи реалізована за допомогою веб-технологій, таких як HTML, CSS, Javascript. Клієнтський додаток працює в браузерах Chrome, Firefox, Safari та Internet Explorer.

Розроблена система відповідає поставленим перед нею вимогам.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1) Saleh, M. S.; Althaibani, A.; Esa, Y.; Mhandi, Y.; Mohamed, A. A. (October 2015). Impact of clustering microgrids on their stability and resilience during blackouts. 2015 International Conference on Smart Grid and Clean Energy Technologies (ICSGCE). pp. 195–200.
2) Torriti, Jacopo (2012). "Demand Side Management for the European Supergrid: Occupancy variances of European single-person households". Energy Policy. 44: 199–206.
3) Energy Plus [Електронний ресурс] – Режим доступу: https://energyplus.net/
4) Open DDS [Електронний ресурс] – Режим доступу: https://opendds.org/
5) Web3 [Електронний ресурс] – Режим доступу: https://web3py.readthedocs.io/en/stable/
6) Ethereum [Електронний ресурс] – Режим доступу: https://ethereum.org/
7) Ethub [Електронний ресурс] – Режим доступу: https://docs.ethhub.io/
8) Unit of work [Електронний ресурс] – Режим доступу: https://www.martinfowler.com/eaaCatalog/unitOfWork.html
9) Repository [Електронний ресурс] – Режим доступу: https://martinfowler.com/eaaCatalog/repository.html
10) Python: [Електронний ресурс] – Режим доступу: https://www.python.org/
11) Solidity [Електронний ресурс] – Режим доступу: https://solidity.readthedocs.io/en/v0.5.12/
12) Javascript: [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/en-US/docs/Web/JavaScript
13) Flask: [Електронний ресурс] – Режим доступу: https://www.palletsprojects.com/p/flask/
14) React: [Електронний ресурс] – Режим доступу: https://reactjs.org/

15) Gunicorn: [Електронний ресурс] – Режим доступу: https://gunicorn.org/
16) Autopep8 [Електронний ресурс] – Режим доступу: https://pypi.org/project/autopep8/
17) Windows [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/windows
18) Linux.org [Електронний ресурс] – Режим доступу: https://www.linux.org/
19) Linux.com [Електронний ресурс] – Режим доступу: https://www.linux.com/
20) SQL Server [Електронний ресурс] – Режим доступу: https://www.microsoft.com/en-us/sql-server/sql-server-2019
21) Nginx.com [Електронний ресурс] – Режим доступу: https://www.nginx.com/
22) Nginx.org [Електронний ресурс] – Режим доступу: https://www.nginx.org/
23) Visual Studio Code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/
24) Remix [Електронний ресурс] – Режим доступу: https://remix.ethereum.org/
25) HTTPS [Електронний ресурс] – Режим доступу: https://www.ssl.com/faqs/what-is-https/
26) RPC [Електронний ресурс] – Режим доступу: https://grpc.io/
27) Git [Електронний ресурс] – Режим доступу: https://git-scm.com/
28) Github [Електронний ресурс] – Режим доступу: https://github.com/
29) Github Actions [Електронний ресурс] – Режим доступу: https://github.com/features/actions
30) DigitalOcean [Електронний ресурс] – Режим доступу: https://www.digitalocean.com/
31) Trello [Електронний ресурс] – Режим доступу: https://trello.com/en
32) Microservices [Електронний ресурс] – Режим доступу: https://martinfowler.com/articles/microservices.html#footnote-etymology

33) Software Architecture Patterns by Mark Richards. Publisher: O'Reilly Media Inc. Release Date: February 2015.

34) CQRS [Електронний ресурс] – Режим доступу: https://martinfowler.com/bliki/CQRS.html

35) SOLID [Електронний ресурс] – Режим доступу: https://clean-code-developer.com/weitere-infos/solid/

36) Erich Gamma; Richard Helm; Ralph Johnson; John Vlissides (1994). Design Patterns: Elements of Reusable Object-Oriented Software. Addison Wesley. pp. 87ff.